RM0533
Reference manual

STM32H7Pxxx Arm®-based 32-bit MCUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32H7Pxxx microprocessor memory and peripherals.

The STM32H7Pxxx is a family of microprocessors with different packages and peripherals.

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

For information on the Cortex®-M7 core, refer to the corresponding Technical Reference Manuals available on http://infocenter.arm.com.

The STM32H7Pxxx microprocessors include ST state-of-the-art patented technology.

Related documents

- Cortex®-M7 Technical Reference Manual, available from: http://infocenter.arm.com
- STM32H7Pxxx data brief (DS15208)
- STM32 Cortex®-M7 programming manual (PM0253)
- STM32H7Pxxx device errata (ES0678)

Contents

1 Documentation conventions 106

1.1 General information 106

1.2 List of abbreviations for registers 106

1.3 Register reset value 107

1.4 Glossary 107

1.5 Availability of peripherals 107

1.6 Availability of security features 107

2 Memory and bus architecture 108

2.1 System architecture 108

2.1.1 Bus matrices 109

2.1.2 Peripheral buses 110

2.1.3 CPU buses 110

2.1.4 Bus manager peripherals 111

2.2 AXI interconnect matrix (AXIM) 111

2.2.1 AXI introduction 111

2.2.2 AXI interconnect main features 112

2.2.3 Block diagram 112

2.2.4 AXI interconnect registers 115

2.2.5 AXI interconnect register map 122

2.3 Memory organization 132

2.3.1 Introduction 132

2.3.2 Memory map and register boundary addresses 133

2.4 Embedded SRAM 141

2.5 Flash memory overview 142

2.6 Boot configuration 143

3 System security 145

3.1 Key security features 145

3.2 Identification 145

3.2.1 Unique ID 145

3.2.2 Discover command 146

3.3 Secure boot 146

3.3.1 Unique boot entry . . . 146
3.3.2 Embedded flash memory write protection . . . 146
3.3.3 Boot levels isolation with HDP . . . 147
3.4 Isolation . . . 147
3.4.1 Temporal isolation using hide protection levels (HDPL) . . . 147
3.4.2 Isolation using privileged mode . . . 147
3.5 Secure storage . . . 149
3.5.1 Boot data storage . . . 149
3.5.2 Battery-powered volatile secure storage . . . 149
3.6 Cryptographic engines . . . 150
3.6.1 Cryptographic random number generation . . . 150
3.6.2 Cryptographic operations . . . 150
3.7 Tamper detection and response . . . 150
3.7.1 Tamper detection sources . . . 150
3.7.2 Response to tampers . . . 151
3.7.3 Software filtering mechanism . . . 152
3.7.4 Tamper detection and low-power modes . . . 152
3.8 Product life cycle . . . 153
3.8.1 Life cycle management . . . 154
3.9 System memory . . . 154
3.9.1 RSS user functions . . . 154
3.9.2 RSSLIB . . . 155

4 RAM ECC monitoring (RAMECC) . . . 162
4.1 Introduction . . . 162
4.2 RAMECC main features . . . 162
4.3 RAMECC functional description . . . 162
4.3.1 RAMECC internal signals . . . 164
4.3.2 RAMECC monitor mapping . . . 165
4.3.3 ECC fault injection test on SRAM1, SRAM3, and TCM . . . 166
4.3.4 Read access latency . . . 166
4.4 RAMECC low-power modes . . . 166
4.5 RAMECC interrupts . . . 167
4.6 RAMECC registers . . . 168
4.6.1 RAMECC interrupt enable register (RAMECC_IER) . . . 168

4.6.2 RAMECC monitor x configuration register (RAMECC_MxCR) . . . 169
4.6.3 RAMECC monitor x status register (RAMECC_MxSR) . . . 170
4.6.4 RAMECC monitor x failing address register (RAMECC_MxFAR) . . . 170
4.6.5 RAMECC monitor x failing data register (RAMECC_MxFDR) . . . 171
4.6.6 RAMECC monitor x failing ECC error code register (RAMECC_MxFECR) . . . 171
4.6.7 RAMECC register map . . . 171

5 Embedded flash memory (FLASH) . . . 175

5.1 Introduction . . . 175
5.2 FLASH main features . . . 175
5.3 FLASH functional description . . . 176
5.3.1 FLASH block diagram . . . 176
5.3.2 FLASH internal signals . . . 177
5.3.3 FLASH architecture and integration in the system . . . 177
5.3.4 Flash memory architecture and usage . . . 179
5.3.5 Flash hide protection schemes . . . 183
5.3.6 Overview of FLASH operations . . . 184
5.3.7 FLASH read operations . . . 185
5.3.8 FLASH program operations . . . 188
5.3.9 FLASH erase operations . . . 192
5.3.10 FLASH parallel operations . . . 193
5.3.11 Flash memory error protections . . . 193
5.3.12 FLASH one-time programmable (OTP) area . . . 195
5.3.13 FLASH read-only (RO) area . . . 197
5.3.14 FLASH bank and register swapping . . . 197
5.3.15 FLASH reset and clocks . . . 199
5.3.16 Tamper management . . . 200
5.4 FLASH option bytes . . . 200
5.4.1 About option bytes . . . 200
5.4.2 Option byte loading . . . 200
5.4.3 Option byte words modification . . . 200
5.4.4 Option byte key management . . . 202
5.4.5 Option byte user words overview . . . 204

5.4.6 Description of user option byte word 206
5.4.7 Description of security option bytes 207
5.5 FLASH protection mechanisms 208
5.5.1 FLASH configuration protection 208
5.5.2 Write protection 209
5.5.3 Life cycle management 210
5.5.4 Hide protection (HDP) 212
5.6 FLASH low-power modes 215
5.6.1 Introduction 215
5.6.2 Managing the FLASH domain switching to Stop or Standby 215
5.7 FLASH error management 215
5.7.1 Write protection error (WRPERRF) 216
5.7.2 Programming sequence error (PGSERRF) 217
5.7.3 Strobe error (STRBERRF) 217
5.7.4 Inconsistency error (INCERRF) 217
5.7.5 Error correction code error (SNECCERRF/DBECCERRF) 218
5.7.6 Read secure error (RDSERRF) 219
5.7.7 CRC read error (CRCRDERRF) 219
5.7.8 Option byte change error (OPTERRF) 219
5.7.9 Option byte loading error (OBLERRF) 220
5.7.10 Option byte key error flag (OBKERRF) 220
5.7.11 Miscellaneous HardFault errors 220
5.8 FLASH interrupts 221
5.9 FLASH registers 223
5.9.1 FLASH access control register (FLASH_ACCR) 223
5.9.2 FLASH access control register (FLASH_ACCR_ALIAS) 223
5.9.3 FLASH bank x control key register (FLASH_xKEYR) 224
5.9.4 FLASH bank x control register (FLASH_xCR) 225
5.9.5 FLASH bank x status register (FLASH_xSR) 227
5.9.6 FLASH bank x interrupt enable register (FLASH_xIER) 228
5.9.7 FLASH bank x interrupt status register (FLASH_xISR) 230
5.9.8 FLASH bank x interrupt clear register (FLASH_xICR) 231
5.9.9 FLASH bank x CRC control register (FLASH_xCRCCR) 232
5.9.10 FLASH bank x CRC start address register (FLASH_xCRCSADDR) . . 234
5.9.11 FLASH bank x CRC end address register (FLASH_xCRCEADDR) . . 235
5.9.12 FLASH CRC data register (FLASH_CRCDATAR) 235

5.9.13 FLASH bank x ECC single error fail address (FLASH_xECCSFADDR) . . . 236
5.9.14 FLASH bank x ECC double error fail address (FLASH_xECCDFADDR) . . . 236
5.9.15 FLASH bank x write page protection status register (FLASH_xWRPSR) . 237
5.9.16 FLASH bank x write page protection status register programming (FLASH_xWRPSRP) . . . 237
5.9.17 FLASH bank x hide protection status register (FLASH_xHDPSR) . . . 238
5.9.18 FLASH bank x hide protection status register programming (FLASH_xHDPSRP) . . . 238
5.9.19 FLASH bank x HDP extension register (FLASH_xHDPEXTR) . . . 239
5.9.20 FLASH options key register (FLASH_OPTKEYR) . . . 240
5.9.21 FLASH options control register (FLASH_OPTCR) . . . 240
5.9.22 FLASH options interrupt status register (FLASH_OPTISR) . . . 241
5.9.23 FLASH options interrupt clear register (FLASH_OPTICR) . . . 242
5.9.24 FLASH OTP lock status register (FLASH_OTPLSR) . . . 242
5.9.25 FLASH OTP lock status register programming (FLASH_OTPLSRP) . 243
5.9.26 FLASH nonvolatile status register (FLASH_NVSR) . . . 243
5.9.27 FLASH security status register programming (FLASH_NVSRP) . . . 244
5.9.28 FLASH RoT status register (FLASH_ROTSR) . . . 244
5.9.29 FLASH RoT status register programming (FLASH_ROTSRP) . . . 245
5.9.30 FLASH option byte key control register (FLASH_OBKCR) . . . 246
5.9.31 FLASH option byte key clear register (FLASH_OBKCLR) . . . 247
5.9.32 FLASH option bytes key data register x (FLASH_OBKDRx) . . . 247
5.9.33 FLASH option byte word 1 status register (FLASH_OBW1SR) . . . 248
5.9.34 FLASH option byte word 1 status register programming (FLASH_OBW1SRP) . . . 249
5.9.35 FLASH register map . . . 251

6 Power control (PWR) . . . 257

6.1 Introduction . . . 257
6.2 PWR main features . . . 257
6.3 PWR block diagram . . . 258
6.3.1 PWR pins and internal signals . . . 258
6.4 Power supplies . . . 260
6.4.1 System supply startup . . . 262
6.4.2 Core domain . . . 265

6.4.3 Backup domain . . . 268
6.4.4 VBAT battery charging . . . 270
6.4.5 Analog supply . . . 270
6.5 Power supply supervision . . . 271
6.5.1 Power-on reset (POR)/power-down reset (PDR) . . . 272
6.5.2 Brownout reset (BOR) . . . 272
6.5.3 Programmable voltage detector (PVD) . . . 273
6.5.4 Analog voltage detector (AVD) . . . 275
6.5.5 Battery voltage thresholds . . . 276
6.5.6 Temperature thresholds . . . 277
6.5.7 VCORE maximum voltage level detector . . . 278
6.6 Power management . . . 279
6.6.1 Operating modes . . . 279
6.6.2 Voltage scaling . . . 280
6.6.3 Power management examples . . . 281
6.7 Low-power modes . . . 285
6.7.1 Slowing down system clocks . . . 286
6.7.2 Controlling peripheral clocks . . . 286
6.7.3 Entering low-power modes . . . 286
6.7.4 Exiting from low-power modes . . . 286
6.7.5 Sleep mode . . . 288
6.7.6 Stop mode . . . 288
6.7.7 Standby mode . . . 291
6.7.8 Monitoring low-power modes . . . 293
6.8 PWR registers . . . 294
6.8.1 PWR control register 1 (PWR_CR1) . . . 294
6.8.2 PWR status register 1 (PWR_SR1) . . . 295
6.8.3 PWR control status register 1 (PWR_CSR1) . . . 297
6.8.4 PWR control status register 2 (PWR_CSR2) . . . 298
6.8.5 PWR CPU control status register 3 (PWR_CSR3) . . . 299
6.8.6 PWR control status register 4 (PWR_CSR4) . . . 300
6.8.7 PWR wakeup clear register (PWR_WKUPCR) . . . 301
6.8.8 PWR wakeup flag register (PWR_WKUPFR) . . . 302
6.8.9 PWR wakeup enable and polarity register (PWR_WKUPEPR) . . . 303
6.8.10 PWR USB Type-C and power delivery register (PWR_UCPDR) . . . 304
6.8.11 PWR apply pull register (PWR_APCR) . . . 304
6.8.12 PWR I3C pull-up control register (PWR_I3CPUCR) . . . 305

6.8.13 PWR XSPI pull-up control register (PWR_XSPIPUCR) 307
6.8.14 PWR XSPI pull-down control register (PWR_XSPIPDCR) 310
6.8.15 PWR register map 312

7 Reset and clock control (RCC) 314
7.1 RCC main features 314
7.2 RCC block diagram 315
7.3 RCC pins and internal signals 316
7.4 RCC reset block functional description 318
7.4.1 Reset from PWR block 318
7.4.2 The system and application resets (sys_rst and nreset) 320
7.4.3 The NRST reset 320
7.4.4 Low-power mode security reset (lpwr_rst) 320
7.4.5 Backup domain reset 321
7.4.6 Coresight debug reset 321
7.4.7 Option bytes loading 322
7.4.8 Peripheral resets 322
7.4.9 Reset coverage summary 322
7.4.10 Reset source identification 324
7.4.11 Power-on and wakeup sequences 324
7.5 RCC clock block functional description 326
7.5.1 Clock naming convention 328
7.5.2 Oscillators description 328
7.5.3 Clock security system (CSS) 338
7.5.4 Clock output generation (MCO1/MCO2) 341
7.5.5 PLL description 343
7.5.6 System clock (**sys_ck**) 349
7.5.7 Clock protection 351
7.5.8 Clock generation in Stop and Standby modes 352
7.5.9 Peripheral clock distribution 354
7.5.10 General clock concept overview 376
7.5.11 Peripheral allocation 378
7.5.12 Peripheral clock gating control 379
7.5.13 CPU and bus matrix clock gating control 382
7.5.14 Low-power emulation modes 383
7.6 RCC interrupts 384

7.7 RCC Programming examples 385
7.7.1 PLL programming procedure 385
7.7.2 Frequency configuration examples 387
7.8 RCC registers 390
7.8.1 RCC source control register (RCC_CR) 390
7.8.2 RCC clock protection register (RCC_CKPROTR) 393
7.8.3 RCC HSI calibration register (RCC_HSICFGR) 394
7.8.4 RCC clock recovery RC register (RCC_CRRCR) 395
7.8.5 RCC CSI calibration register (RCC_CSICFGR) 395
7.8.6 RCC clock configuration register (RCC_CFGR) 396
7.8.7 RCC CPU domain clock configuration register (RCC_CDCFGR) 398
7.8.8 RCC AHB clock configuration register (RCC_BMCFGR) 399
7.8.9 RCC APB clocks configuration register (RCC_APBCFGR) 400
7.8.10 RCC PLLs clock source selection register (RCC_PLLCKSELR) 401
7.8.11 RCC PLLs configuration register (RCC_PLLCFGR) 402
7.8.12 RCC PLL1 dividers configuration register 1 (RCC_PLL1DIVR1) 406
7.8.13 RCC PLL1 dividers configuration register 2 (RCC_PLL1DIVR2) 407
7.8.14 RCC PLL1 fractional divider register (RCC_PLL1FRACR) 408
7.8.15 RCC PLL1 Spread Spectrum Clock Generator register (RCC_PLL1SSCGR) 409
7.8.16 RCC PLL2 dividers configuration register 1 (RCC_PLL2DIVR1) 410
7.8.17 RCC PLL2 dividers configuration register 2 (RCC_PLL2DIVR2) 411
7.8.18 RCC PLL2 fractional divider register (RCC_PLL2FRACR) 412
7.8.19 RCC PLL2 spread spectrum clock generator register (RCC_PLL2SSCGR) 413
7.8.20 RCC PLL3 dividers configuration register 1 (RCC_PLL3DIVR1) 414
7.8.21 RCC PLL3 dividers configuration register 2 (RCC_PLL3DIVR2) 415
7.8.22 RCC PLL3 fractional divider register (RCC_PLL3FRACR) 416
7.8.23 RCC PLL3 spread spectrum clock generator register (RCC_PLL3SSCGR) 417
7.8.24 RCC peripheral kernel clock selection register 1 (RCC_CCIPR1) . . . 418
7.8.25 RCC peripheral kernel clock selection register 2 . . . . (RCC_CCIPR2) 420
7.8.26 RCC peripheral kernel clock selection register 3 (RCC_CCIPR3) . . . 421
7.8.27 RCC peripheral kernel clock selection register 4 . . . . (RCC_CCIPR4) 422
7.8.28 RCC clock source interrupt enable register (RCC_CIER) 425
7.8.29 RCC clock source interrupt flag register (RCC_CIFR) 426
7.8.30 RCC clock source interrupt clear register (RCC_CICR) 428
7.8.31 RCC Backup domain control register (RCC_BDCR) 430

7.8.32 RCC clock control and status register (RCC_CSR) 432
7.8.33 RCC AHB1 peripheral reset register (RCC_AHB1RSTR) 433
7.8.34 RCC AHB2 peripheral reset register (RCC_AHB2RSTR) 433
7.8.35 RCC AHB3 peripheral reset register (RCC_AHB3RSTR) 434
7.8.36 RCC AHB4 peripheral reset register (RCC_AHB4RSTR) 435
7.8.37 RCC AHB5 peripheral reset register (RCC_AHB5RSTR) 436
7.8.38 RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1) 437
7.8.39 RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2) 440
7.8.40 RCC APB2 peripheral reset register (RCC_APB2RSTR) 441
7.8.41 RCC APB4 peripheral reset register (RCC_APB4RSTR) 443
7.8.42 RCC AXI clocks gating disable register (RCC_CKGDISR) 444
7.8.43 RCC Reset status register (RCC_RSR) 447
7.8.44 RCC AHB1 clock enable register (RCC_AHB1ENR) 448
7.8.45 RCC AHB2 clock enable register (RCC_AHB2ENR) 449
7.8.46 RCC AHB3 clock enable register (RCC_AHB3ENR) 450
7.8.47 RCC AHB4 clock enable register (RCC_AHB4ENR) 451
7.8.48 RCC AHB5 clock enable register (RCC_AHB5ENR) 452
7.8.49 RCC APB1 clock enable register 1 (RCC_APB1ENR1) 453
7.8.50 RCC APB1 clock enable register 2 (RCC_APB1ENR2) 456
7.8.51 RCC APB2 clock enable register (RCC_APB2ENR) 457
7.8.52 RCC APB4 clock enable register (RCC_APB4ENR) 459
7.8.53 RCC AHB5 low-power clock enable register (RCC_AHB5LPENR) . . . 461
7.8.54 RCC AHB1 low-power clock enable register (RCC_AHB1LPENR) . . . 462
7.8.55 RCC AHB2 low-power clock enable register (RCC_AHB2LPENR) . . . 463
7.8.56 RCC AHB3 low-power clock enable register (RCC_AHB3LPENR) . . . 464
7.8.57 RCC AHB4 low-power clock enable register (RCC_AHB4LPENR) . . . 465
7.8.58 RCC APB1 low-power clock enable register 1 (RCC_APB1LPENR1) 466
7.8.59 RCC APB1 low-power clock enable register 2 (RCC_APB1LPENR2) 469
7.8.60 RCC APB2 low-power clock enable register (RCC_APB2LPENR) . . . 470
7.8.61 RCC APB4 low-power clock enable register (RCC_APB4LPENR) . . . 472
7.9 RCC register map 475

8 Clock recovery system (CRS) 483
8.1 CRS introduction 483
8.2 CRS main features 483
8.3 CRS implementation 483
8.4 CRS functional description 484

8.4.1 CRS block diagram . . . 484
8.4.2 CRS internal signals . . . 484
8.4.3 Synchronization input . . . 485
8.4.4 Frequency error measurement . . . 485
8.4.5 Frequency error evaluation and automatic trimming . . . 486
8.4.6 CRS initialization and configuration . . . 487
8.5 CRS in low-power modes . . . 488
8.6 CRS interrupts . . . 488
8.7 CRS registers . . . 488
8.7.1 CRS control register (CRS_CR) . . . 488
8.7.2 CRS configuration register (CRS_CFGR) . . . 490
8.7.3 CRS interrupt and status register (CRS_ISR) . . . 491
8.7.4 CRS interrupt flag clear register (CRS_ICR) . . . 493
8.7.5 CRS register map . . . 493

9 General-purpose I/Os (GPIO) . . . 495
9.1 Introduction . . . 495
9.2 GPIO main features . . . 495
9.3 GPIO functional description . . . 495
9.3.1 General-purpose I/O (GPIO) . . . 497
9.3.2 I/O pin alternate function multiplexer and mapping . . . 498
9.3.3 I/O port control registers . . . 498
9.3.4 I/O port data registers . . . 499
9.3.5 I/O data bitwise handling . . . 499
9.3.6 GPIO locking mechanism . . . 499
9.3.7 I/O alternate function input/output . . . 499
9.3.8 External interrupt/wake-up lines . . . 500
9.3.9 Input configuration . . . 500
9.3.10 Output configuration . . . 501
9.3.11 Alternate function configuration . . . 501
9.3.12 Analog configuration . . . 502
9.3.13 Using the HSE or LSE oscillator pins as GPIOs . . . 502
9.3.14 Using the GPIO pins in the RTC supply domain . . . 503
9.3.15 Privileged and unprivileged modes . . . 503
9.3.16 High-speed low-voltage mode (HSLV) . . . 503
9.3.17 I/O compensation cell . . . 503

9.4 GPIO registers 503
9.4.1 GPIO port mode register . . . . . .(GPIOx_MODER) (x = A to I) 503
9.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to I) 505
9.4.3 GPIO port output speed register . . . (GPIOx_OSPEEDR) (x = A to I) 505
9.4.4 GPIO port pull-up/pull-down register . . . (GPIOx_PUPDR) (x = A to I) 506
9.4.5 GPIO port input data register (GPIOx_IDR) (x = A to I) 506
9.4.6 GPIO port output data register . . . . . (GPIOx_ODR) (x = A to I) 507
9.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to I) 507
9.4.8 GPIO port configuration lock register . . . . (GPIOx_LCKR) (x = A to I) 508
9.4.9 GPIO alternate function low register . . . . .(GPIOx_AFRL) (x = A to I) 509
9.4.10 GPIO alternate function high register . . . (GPIOx_AFRH) (x = A to I) 510
9.4.11 GPIO port bit reset register . . . . . .(GPIOx_BRR) (x = A to I) 511
9.4.12 GPIO register map 512

10 System configuration, boot and security (SBS) 514

10.1 SBS introduction 514
10.2 SBS main features 514
10.3 SBS functional description 515
10.3.1 SBS block diagram 515
10.3.2 SBS internal signals 515
10.3.3 SBS reset and clocks 515
10.3.4 SBS hide protection management 516
10.3.5 SBS boot control 516
10.3.6 SBS debug management 517
10.3.7 SBS voltage booster for I/O analog switches 518
10.3.8 SBS I/O compensation cell management 518
10.3.9 SBS registers access control 519
10.3.10 SBS error management 520
10.4 SBS interrupts 520
10.5 SBS registers 520
10.5.1 SBS boot status register (SBS_BOOTSR) 520
10.5.2 SBS hide protection control register (SBS_HDPLCR) 520
10.5.3 SBS hide protection status register (SBS_HDPLSR) 521
10.5.4 SBS debug control register (SBS_DBGCR) 521
10.5.5 SBS debug lock register (SBS_DBGLOCKR) 522
10.5.6 SBS RSS command register (SBS_RSSCMDR) 522

10.5.7 SBS product mode and configuration register (SBS_PMCR) 523
10.5.8 SBS FPU interrupt mask register (SBS_FPUIMR) 524
10.5.9 SBS memory erase status register (SBS_MESR) 524
10.5.10 SBS I/O compensation cell control and status register (SBS_CCCSR) 525
10.5.11 SBS compensation cell for I/Os value register (SBS_CCVALR) 526
10.5.12 SBS compensation cell for I/Os software value register (SBS_CCSWVALR) 526
10.5.13 SBS break lock register (SBS_BKLOCKR) 527
10.5.14 SBS external interrupt configuration register x (SBS_EXTICRx) 529
10.5.15 SBS register map 531

11 Peripheral interconnect 533
11.1 Introduction 533
11.2 Connection summary 533
11.3 Interconnection details 536
11.3.1 Master to slave interconnection for timers 536
11.3.2 Triggers to ADCs 537
11.3.3 ADC analog watchdogs as triggers to timers 538
11.3.4 Comparator outputs to timers 538
11.3.5 HSP break to PWM timers 540
11.3.6 Triggers on ADF1 540
11.3.7 Internal input and clock sources to timers 541
11.3.8 Triggers to low-power timers 542
11.3.9 RTC wake-up as inputs to timers 542
11.3.10 OTG_FS SOF as trigger to timers 542
11.3.11 System errors as break signals to timers 543
11.3.12 Triggers to GPDMA/HPDMA 543
11.3.13 Internal analog signals to analog peripherals 544
11.3.14 ADC data filtering by the ADF1 545
11.3.15 Internal tamper sources 545
11.3.16 Output from tamper to RTC 545
11.3.17 I3C in-band interrupt acknowledge to timers 545
11.3.18 NPU input triggers 546
11.3.19 HSP input triggers 546

12 High-performance direct memory access controller (HPDMA) 548

12.1 HPDMA introduction . . . 548
12.2 HPDMA main features . . . 548
12.3 HPDMA implementation . . . 549
12.3.1 HPDMA channels . . . 549
12.3.2 HPDMA in low-power modes . . . 550
12.3.3 HPDMA requests . . . 550
12.3.4 HPDMA block requests . . . 554
12.3.5 HPDMA channels with peripheral early termination . . . 554
12.3.6 HPDMA triggers . . . 555
12.4 HPDMA functional description . . . 557
12.4.1 HPDMA block diagram . . . 557
12.4.2 HPDMA channel state and direct programming without any linked-list . . . 558
12.4.3 HPDMA channel suspend and resume . . . 559
12.4.4 HPDMA channel abort and restart . . . 560
12.4.5 HPDMA channel terminate and drain FIFO . . . 561
12.4.6 HPDMA linked-list data structure . . . 563
12.4.7 Linked-list item transfer execution . . . 566
12.4.8 HPDMA channel state and linked-list programming in run-to-completion mode . . . 566
12.4.9 HPDMA channel state and linked-list programming in link step mode . . . 570
12.4.10 HPDMA channel state and linked-list programming . . . 577
12.4.11 HPDMA FIFO-based transfers . . . 579
12.4.12 HPDMA transfer request and arbitration . . . 595
12.4.13 HPDMA triggered transfer . . . 599
12.4.14 HPDMA circular buffering with linked-list programming . . . 600
12.4.15 HPDMA transfer in peripheral flow-control mode . . . 602
12.4.16 HPDMA byte-based padding at the block level . . . 603
12.4.17 HPDMA privileged/unprivileged channel . . . 604
12.4.18 HPDMA error management . . . 605
12.5 HPDMA in debug mode . . . 606
12.6 HPDMA in low-power modes . . . 607
12.7 HPDMA interrupts . . . 607
12.8 HPDMA registers . . . 609
12.8.1 HPDMA privileged configuration register (HPDMA_PRIVCFGR) . . . 609
12.8.2 HPDMA configuration lock register (HPDMA_RCFGLOCKR) . . . 609

12.8.3 HPDMA masked interrupt status register (HPDMA_MISR) 610
12.8.4 HPDMA channel x linked-list base address register (HPDMA_CxLBAR) 611
12.8.5 HPDMA channel x flag clear register (HPDMA_CxFCR) 611
12.8.6 HPDMA channel x status register (HPDMA_CxSR) 612
12.8.7 HPDMA channel x control register (HPDMA_CxCR) 614
12.8.8 HPDMA channel x transfer register 1 (HPDMA_CxTR1) 616
12.8.9 HPDMA channel x transfer register 2 (HPDMA_CxTR2) 620
12.8.10 HPDMA channel x block register 1 (HPDMA_CxBR1) 624
12.8.11 HPDMA channel x alternate block register 1 (HPDMA_CxBR1) 625
12.8.12 HPDMA channel x source address register (HPDMA_CxSAR) 627
12.8.13 HPDMA channel x destination address register (HPDMA_CxDAR) . . 629
12.8.14 HPDMA channel x transfer register 3 (HPDMA_CxTR3) 630
12.8.15 HPDMA channel x block register 2 (HPDMA_CxBR2) 631
12.8.16 HPDMA channel x linked-list address register (HPDMA_CxLLR) 632
12.8.17 HPDMA channel x alternate linked-list address register (HPDMA_CxLLR) 634
12.8.18 HPDMA register map 635

13 General purpose direct memory access controller (GPDMA) 638

13.1 GPDMA introduction 638
13.2 GPDMA main features 638
13.3 GPDMA implementation 639
13.3.1 GPDMA channels 639
13.3.2 GPDMA in low-power modes 640
13.3.3 GPDMA requests 640
13.3.4 GPDMA block requests 644
13.3.5 GPDMA channels with peripheral early termination 645
13.3.6 GPDMA triggers 645
13.4 GPDMA functional description 648
13.4.1 GPDMA block diagram 648
13.4.2 GPDMA channel state and direct programming without any linked-list 648
13.4.3 GPDMA channel suspend and resume 649
13.4.4 GPDMA channel abort and restart 650
13.4.5 GPDMA channel terminate and drain FIFO 651
13.4.6 GPDMA linked-list data structure 652
13.4.7 Linked-list item transfer execution 656

13.4.8 GPDMA channel state and linked-list programming in run-to-completion mode . . . 656
13.4.9 GPDMA channel state and linked-list programming in link step mode . . . 660
13.4.10 GPDMA channel state and linked-list programming . . . 667
13.4.11 GPDMA FIFO-based transfers . . . 669
13.4.12 GPDMA transfer request and arbitration . . . 676
13.4.13 GPDMA triggered transfer . . . 680
13.4.14 GPDMA circular buffering with linked-list programming . . . 681
13.4.15 GPDMA transfer in peripheral flow-control mode . . . 683
13.4.16 GPDMA byte-based padding at the block level . . . 684
13.4.17 GPDMA privileged/unprivileged channel . . . 685
13.4.18 GPDMA error management . . . 686
13.5 GPDMA in debug mode . . . 687
13.6 GPDMA in low-power modes . . . 688
13.7 GPDMA interrupts . . . 688
13.8 GPDMA registers . . . 690
13.8.1 GPDMA privileged configuration register (GPDMA_PRIVCFGR) . . . 690
13.8.2 GPDMA configuration lock register (GPDMA_RCFGLOCKR) . . . 690
13.8.3 GPDMA masked interrupt status register (GPDMA_MISR) . . . 691
13.8.4 GPDMA channel x linked-list base address register (GPDMA_CxLBAR) . . . 691
13.8.5 GPDMA channel x flag clear register (GPDMA_CxFCR) . . . 692
13.8.6 GPDMA channel x status register (GPDMA_CxSR) . . . 693
13.8.7 GPDMA channel x control register (GPDMA_CxCR) . . . 695
13.8.8 GPDMA channel x transfer register 1 (GPDMA_CxTR1) . . . 697
13.8.9 GPDMA channel x transfer register 2 (GPDMA_CxTR2) . . . 700
13.8.10 GPDMA channel x block register 1 (GPDMA_CxBR1) . . . 704
13.8.11 GPDMA channel x alternate block register 1 (GPDMA_CxBR1) . . . 705
13.8.12 GPDMA channel x source address register (GPDMA_CxSAR) . . . 707
13.8.13 GPDMA channel x destination address register (GPDMA_CxDAR) . . 709
13.8.14 GPDMA channel x transfer register 3 (GPDMA_CxTR3) . . . 710
13.8.15 GPDMA channel x block register 2 (GPDMA_CxBR2) . . . 711
13.8.16 GPDMA channel x linked-list address register (GPDMA_CxLLR) . . . 712
13.8.17 GPDMA channel x alternate linked-list address register (GPDMA_CxLLR) . . . 714
13.8.18 GPDMA register map . . . 715

14 Neural-ART accelerator™ (NPU) 718
14.1 NPU introduction 718
14.2 NPU implementation 719
14.3 NPU functional description 720
14.3.1 NPU block diagram 720
14.3.2 NPU pins and internal signals 720
14.3.3 Configuration network 721
14.3.4 Clock/reset manager 721
14.3.5 Stream link 721
14.3.6 Stream switch 722
14.3.7 Stream engines 722
14.3.8 Encryption/Decryption unit 723
14.3.9 Decompression unit 723
14.3.10 Convolutional accelerator 724
14.3.11 Pooling unit 725
14.3.12 Activation unit 726
14.3.13 Arithmetic unit 729
14.3.14 Reconfigurable buffer 730
14.3.15 Epoch controller 730
14.3.16 Interrupt controller 731
14.4 Configuration examples 731
14.4.1 DMA transfer 731
14.4.2 Simple processing 731
14.4.3 Multiple processing 732
14.4.4 Conv-Pool-ReLU 732
14.5 Address space 733

15 Nested vectored interrupt controller (NVIC) 734
15.1 NVIC features 734
15.1.1 SysTick 734
15.1.2 Interrupt and exception vectors 736

16 Extended interrupt and event controller (EXTI) 744
16.1 EXTI main features 744
16.2 EXTI block diagram 744
16.2.1 EXTI connections between peripherals and CPU 745

16.3 EXTI functional description 746
16.3.1 EXTI configurable event input - CPU wake-up 747
16.3.2 EXTI direct event input - CPU wake up 748
16.4 EXTI event input mapping 749
16.5 EXTI functional behavior 751
16.5.1 EXTI CPU interrupt procedure 752
16.5.2 EXTI CPU event procedure 752
16.5.3 EXTI CPU wake-up procedure 752
16.5.4 EXTI software interrupt/event trigger procedure 752
16.6 EXTI registers 753
16.6.1 EXTI rising trigger selection register 1 (EXTI_RTSR1) 753
16.6.2 EXTI falling trigger selection register 1 (EXTI_FTSR1) 753
16.6.3 EXTI software interrupt event register 1 (EXTI_SWIER1) 754
16.6.4 EXTI rising trigger selection register 2 (EXTI_RTSR2) 754
16.6.5 EXTI falling trigger selection register 2 (EXTI_FTSR2) 755
16.6.6 EXTI software interrupt event register 2 (EXTI_SWIER2) 756
16.6.7 EXTI rising trigger selection register 3 (EXTI_RTSR3) 757
16.6.8 EXTI falling trigger selection register 3 (EXTI_FTSR3) 757
16.6.9 EXTI software interrupt event register 3 (EXTI_SWIER3) 758
16.6.10 EXTI interrupt mask register 1 (EXTI_IMR1) 758
16.6.11 EXTI event mask register 1 (EXTI_EMR1) 758
16.6.12 EXTI pending register 1 (EXTI_PR1) 759
16.6.13 EXTI interrupt mask register 2 (EXTI_IMR2) 760
16.6.14 EXTI event mask register 2 (EXTI_EMR2) 760
16.6.15 EXTI pending register 2 (EXTI_PR2) 761
16.6.16 EXTI interrupt mask register 3 (EXTI_IMR3) 762
16.6.17 EXTI event mask register 3 (EXTI_EMR3) 762
16.6.18 EXTI pending register 3 (EXTI_PR3) 763
16.6.19 EXTI register map 764

17 Cyclic redundancy check calculation unit (CRC) 766
17.1 CRC introduction 766
17.2 CRC main features 766
17.3 CRC functional description 767
17.3.1 CRC block diagram 767
17.3.2 CRC internal signals 767

17.3.3 CRC operation . . . 767
17.4 CRC registers . . . 769
17.4.1 CRC data register (CRC_DR) . . . 769
17.4.2 CRC independent data register (CRC_IDR) . . . 769
17.4.3 CRC control register (CRC_CR) . . . 770
17.4.4 CRC initial value (CRC_INIT) . . . 771
17.4.5 CRC polynomial (CRC_POL) . . . 771
17.4.6 CRC register map . . . 772

18 CORDIC coprocessor (CORDIC) . . . 773
18.1 CORDIC introduction . . . 773
18.2 CORDIC main features . . . 773
18.3 CORDIC functional description . . . 773
18.3.1 General description . . . 773
18.3.2 CORDIC functions . . . 773
18.3.3 Fixed point representation . . . 780
18.3.4 Scaling factor . . . 780
18.3.5 Precision . . . 781
18.3.6 Zero-overhead mode . . . 784
18.3.7 Polling mode . . . 785
18.3.8 Interrupt mode . . . 786
18.3.9 DMA mode . . . 786
18.4 CORDIC registers . . . 787
18.4.1 CORDIC control/status register (CORDIC_CSR) . . . 787
18.4.2 CORDIC argument register (CORDIC_WDATA) . . . 789
18.4.3 CORDIC result register (CORDIC_RDATA) . . . 790
18.4.4 CORDIC register map . . . 790

19 Filter math accelerator (FMAC) . . . 791
19.1 FMAC introduction . . . 791
19.2 FMAC main features . . . 791
19.3 FMAC functional description . . . 792
19.3.1 General description . . . 792
19.3.2 Local memory and buffers . . . 793
19.3.3 Input buffers . . . 793
19.3.4 Output buffer . . . 796

19.3.5 Initialization functions . . . 798
19.3.6 Filter functions . . . 799
19.3.7 Fixed point representation . . . 803
19.3.8 Implementing FIR filters with the FMAC . . . 803
19.3.9 Implementing IIR filters with the FMAC . . . 805
19.3.10 Examples of filter initialization . . . 807
19.3.11 Examples of filter operation . . . 808
19.3.12 Filter design tips . . . 810
19.4 FMAC registers . . . 811
19.4.1 FMAC X1 buffer configuration register (FMAC_X1BUFCFG) . . . 811
19.4.2 FMAC X2 buffer configuration register (FMAC_X2BUFCFG) . . . 811
19.4.3 FMAC Y buffer configuration register (FMAC_YBUFCFG) . . . 812
19.4.4 FMAC parameter register (FMAC_PARAM) . . . 813
19.4.5 FMAC control register (FMAC_CR) . . . 814
19.4.6 FMAC status register (FMAC_SR) . . . 815
19.4.7 FMAC write data register (FMAC_WDATA) . . . 816
19.4.8 FMAC read data register (FMAC_RDATA) . . . 817
19.4.9 FMAC register map . . . 817

20 Hardware signal processor (HSP) . . . 819
20.1 HSP introduction . . . 819
20.2 HSP main features . . . 821
20.3 HSP implementation . . . 822
20.4 HSP functional description . . . 823
20.4.1 HSP block diagram . . . 823
20.4.2 HSP pins and internal signals . . . 823
20.4.3 Floating-point format . . . 826
20.4.4 Memory management . . . 827
20.4.5 HSP boot . . . 831
20.4.6 Message box (MSGB) . . . 835
20.4.7 Direct command interface (DCMD) . . . 837
20.4.8 STREAM interface . . . 841
20.4.9 Trigger input interface (TRGITF) . . . 845
20.4.10 Software events generator (SEG) . . . 848
20.4.11 Event controller (EVTC) . . . 851
20.4.12 HSP interrupt controller (HITC) . . . 859
20.4.13 HSP reset and clocking . . . 867

20.4.14 Task comparator unit (TCU) . . . 869
20.4.15 Trigger outputs (TRGO) . . . 871
20.4.16 Time-stamp capture (TSC) . . . 872
20.4.17 BRAM conflict counter . . . 879
20.4.18 Snoop function (SNOOP) . . . 879
20.4.19 Break interface (BKITF) . . . 880
20.5 Debug . . . 884
20.5.1 Register protection . . . 885
20.6 HSP firmware description . . . 887
20.6.1 Processing list mapping . . . 887
20.6.2 Sending commands to the HSP . . . 888
20.6.3 Recording processing lists . . . 891
20.6.4 Executing direct commands (accelerator mode) . . . 892
20.6.5 Data buffer handling . . . 894
20.6.6 Performance monitoring . . . 894
20.6.7 Processing functions summary . . . 894
20.7 HSP registers . . . 900
20.7.1 HSP control register (HSP_CR) . . . 900
20.7.2 HSP CPU-to-HSP semaphore register (HSP_C2HSEMR) . . . 902
20.7.3 HSP CPU-to-HSP message data register (HSP_C2HMSGDR) . . . 902
20.7.4 HSP HSP-to-CPU semaphore register (HSP_H2CSEMR) . . . 903
20.7.5 HSP HSP-to-CPU message data register (HSP_H2CMSGDR) . . . 903
20.7.6 HSP DCMD command status register (HSP_DCMDSR) . . . 904
20.7.7 HSP DCMD pointer status register (HSP_DCMDPTSR) . . . 904
20.7.8 HSP DCMD command ID register (HSP_DCMDIDR) . . . 905
20.7.9 HSP DCMD address pointer register x (HSP_DCMDPTRx) . . . 906
20.7.10 HSP event enable register (HSP_EVTENR) . . . 906
20.7.11 HSP event synchronization enable register (HSP_ESYNCENR) . . . 907
20.7.12 HSP event synchronization source register 0 (HSP_ESYNC0SRCR) . . . 907
20.7.13 HSP event synchronization source register 1 (HSP_ESYNC1SRCR) . . . 909
20.7.14 HSP event synchronization source register 2 (HSP_ESYNC2SRCR) . . . 911
20.7.15 HSP pending event level register (HSP_PEVTLR) . . . 913
20.7.16 HSP interface enable register (HSP_ITFENR) . . . 913
20.7.17 HSP event source register 0 (HSP_EVTSRC0R) . . . 915
20.7.18 HSP event source register 1 (HSP_EVTSRC1R) . . . 917

20.7.19 HSP event source register 2 (HSP_EVTSRC2R) 920
20.7.20 HSP BUFF configuration register (HSP_BUFFCFGR) 922
20.7.21 HSP BUFFx data register x (HSP_BUFFxDR) 923
20.7.22 HSP TRGIN configuration register (HSP_TRGINCFGR) 923
20.7.23 HSP TRGO configuration register (HSP_TRGOCFGR) 924
20.7.24 HSP TRGIN input selection register 0 (HSP_TRGINSELR0) 924
20.7.25 HSP TRGIN input selection register 1 (HSP_TRGINSELR1) 925
20.7.26 HSP TRGIN input selection register 2 (HSP_TRGINSELR2) 926
20.7.27 HSP CPU shared event generator register (HSP_CSEGR) 927
20.7.28 HSP CPU dedicated event generation register (HSP_CDEGR) 928
20.7.29 HSP task comparator unit register (HSP_TCUCFGR) 928
20.7.30 HSP task overlap control register (HSP_TOVLPCR) 930
20.7.31 HSP SNOOP register (HSP_SNPR) 930
20.7.32 HSP conflict counter register (HSP_CCNTR) 931
20.7.33 HSP data capture register (HSP_CAPDR) 932
20.7.34 HSP data capture register [alternate] (HSP_CAPDR) 932
20.7.35 HSP capture control register (HSP_CAPCR) 933
20.7.36 HSP break output configuration register x (HSP_BKOxCFGR) 934
20.7.37 HSP break input configuration register (HSP_BKICFGR) 935
20.7.38 HSP SPE interrupt enable register (HSP_SPE_IER) 937
20.7.39 HSP SPE interrupt status register (HSP_SPE_ISR) 937
20.7.40 HSP error interrupt enable register (HSP_ERR_IER) 938
20.7.41 HSP events interrupt enable register (HSP_EVT_IER) 940
20.7.42 HSP processing event interrupt enable register (HSP_PFCTEVT_IER) 941
20.7.43 HSP error interrupt status register (HSP_ERR_ISR) 942
20.7.44 HSP events interrupt status register (HSP_EVT_ISR) 943
20.7.45 HSP processing event interrupt status register (HSP_PFCTEVT_ISR) 945
20.7.46 HSP error interrupt clear register (HSP_ERR_ICR) 945
20.7.47 HSP events interrupt clear register (HSP_EVT_ICR) 947
20.7.48 HSP processing event interrupt clear register (HSP_PFCTEVT_ICR) 948
20.7.49 HSP error information register (HSP_ERRINFR) 948
20.7.50 HSP firmware error register (HSP_FWERR) 950
20.7.51 HSP parameter register x (HSP_PARAMRx) 950
20.7.52 HSP firmware versions register (HSP_FWVERR) 951
20.7.53 HSP SPE version register (HSP_SPEVERR) 951
20.7.54 HSP register map 952

21 Flexible memory controller (FMC) 958
21.1 FMC main features 958
21.2 FMC block diagram 959
21.3 FMC internal signals 961
21.4 AHB interface 961
21.5 AXI interface 961
21.5.1 Supported memories and transactions 962
21.6 External device address mapping 963
21.6.1 NOR/PSRAM address mapping 964
21.6.2 NAND flash memory address mapping 964
21.6.3 SDRAM address mapping 965
21.7 NOR flash/PSRAM controller 968
21.7.1 External memory interface signals 970
21.7.2 Supported memories and transactions 972
21.7.3 General timing rules 973
21.7.4 NOR flash/PSRAM controller asynchronous transactions 973
21.7.5 Synchronous transactions 993
21.7.6 NOR/PSRAM controller registers 999
21.8 NAND flash controller 1008
21.8.1 External memory interface signals 1008
21.8.2 NAND flash supported memories and transactions 1009
21.8.3 Timing diagrams for NAND flash memories 1010
21.8.4 NAND flash operations 1011
21.8.5 NAND flash prewait feature 1012
21.8.6 Computation of the error correction code (ECC) in NAND flash memory 1013
21.8.7 NAND flash controller registers 1014
21.9 SDRAM controller 1020
21.9.1 SDRAM controller main features 1020
21.9.2 SDRAM External memory interface signals 1020
21.9.3 SDRAM controller functional description 1021
21.9.4 Low-power modes 1027
21.9.5 SDRAM controller registers 1029
21.9.6 FMC register map 1036

22 Extended-SPI interface (XSPI) 1039

22.1 XSPI introduction . . . 1039
22.2 XSPI main features . . . 1039
22.3 XSPI implementation . . . 1040
22.4 XSPI functional description . . . 1041
22.4.1 XSPI block diagram . . . 1041
22.4.2 XSPI pins and internal signals . . . 1042
22.4.3 Clock constraints . . . 1043
22.4.4 XSPI interface to memory modes . . . 1043
22.4.5 XSPI regular-command protocol . . . 1043
22.4.6 XSPI regular-command protocol signal interface . . . 1047
22.4.7 HyperBus protocol . . . 1050
22.4.8 Specific features . . . 1054
22.4.9 XSPI operating mode introduction . . . 1055
22.4.10 XSPI indirect mode . . . 1055
22.4.11 XSPI automatic status-polling mode . . . 1057
22.4.12 XSPI memory-mapped mode . . . 1057
22.4.13 XSPI configuration introduction . . . 1059
22.4.14 XSPI system configuration . . . 1059
22.4.15 XSPI device configuration . . . 1059
22.4.16 XSPI regular-command mode configuration . . . 1062
22.4.17 XSPI HyperBus protocol configuration . . . 1065
22.4.18 XSPI error management . . . 1065
22.4.19 XSPI BUSY and ABORT . . . 1066
22.4.20 XSPI reconfiguration or deactivation . . . 1067
22.4.21 NCS behavior . . . 1067
22.4.22 Software control of two external memories . . . 1067
22.4.23 Hardware-controlled extended memory support . . . 1068
22.5 Address alignment and data number . . . 1069
22.6 XSPI interrupts . . . 1070
22.7 XSPI registers . . . 1071
22.7.1 XSPI control register (XSPI_CR) . . . 1071
22.7.2 XSPI device configuration register 1 (XSPI_DCR1) . . . 1074
22.7.3 XSPI device configuration register 2 (XSPI_DCR2) . . . 1076
22.7.4 XSPI device configuration register 3 (XSPI_DCR3) . . . 1077
22.7.5 XSPI device configuration register 4 (XSPI_DCR4) . . . 1077
22.7.6 XSPI status register (XSPI_SR) . . . 1078

22.7.7 XSPI flag clear register (XSPI_FCR) 1079
22.7.8 XSPI data length register (XSPI_DLR) 1079
22.7.9 XSPI address register (XSPI_AR) 1080
22.7.10 XSPI data register (XSPI_DR) 1081
22.7.11 XSPI polling status mask register (XSPI_PSMKR) 1081
22.7.12 XSPI polling status match register (XSPI_PSMAR) 1082
22.7.13 XSPI polling interval register (XSPI_PIR) 1082
22.7.14 XSPI communication configuration register (XSPI_CCR) 1082
22.7.15 XSPI timing configuration register (XSPI_TCR) 1084
22.7.16 XSPI instruction register (XSPI_IR) 1085
22.7.17 XSPI alternate bytes register (XSPI_ABR) 1085
22.7.18 XSPI low-power timeout register (XSPI_LPTR) 1086
22.7.19 XSPI wrap communication configuration register (XSPI_WPCCR) 1086
22.7.20 XSPI wrap timing configuration register (XSPI_WPTCR) 1088
22.7.21 XSPI wrap instruction register (XSPI_WPIR) 1089
22.7.22 XSPI wrap alternate bytes register (XSPI_WPABR) 1089
22.7.23 XSPI write communication configuration register (XSPI_WCCR) 1090
22.7.24 XSPI write timing configuration register (XSPI_WTCR) 1092
22.7.25 XSPI write instruction register (XSPI_WIR) 1092
22.7.26 XSPI write alternate bytes register (XSPI_WABR) 1093
22.7.27 XSPI HyperBus latency configuration register (XSPI_HLCR) 1093
22.7.28 XSPI register map 1094

23 Secure digital input/output MultiMediaCard interface (SDMMC) . . 1097

23.1 SDMMC main features 1097
23.2 SDMMC implementation 1097
23.3 SDMMC bus topology 1098
23.4 SDMMC operation modes 1100
23.5 SDMMC functional description 1101
23.5.1 SDMMC block diagram 1101
23.5.2 SDMMC pins and internal signals 1101
23.5.3 General description 1102
23.5.4 SDMMC adapter 1104
23.5.5 SDMMC AHB slave interface 1126

23.5.6 SDMMC AHB master interface 1127
23.5.7 AHB and SDMMC_CK clock relation 1130
23.6 Card functional description 1130
23.6.1 SD I/O mode 1130
23.6.2 CMD12 send timing 1138
23.6.3 Sleep (CMD5) 1142
23.6.4 Interrupt mode (Wait-IRQ) 1143
23.6.5 Boot operation 1144
23.6.6 Response R1b handling 1147
23.6.7 Reset and card cycle power 1148
23.7 Hardware flow control 1149
23.8 Ultra-high-speed phase I (UHS-I) voltage switch 1150
23.9 SDMMC interrupts 1153
23.10 SDMMC registers 1154
23.10.1 SDMMC power control register (SDMMC_POWER) 1154
23.10.2 SDMMC clock control register (SDMMC_CLKCR) 1155
23.10.3 SDMMC argument register (SDMMC_ARGR) 1157
23.10.4 SDMMC command register (SDMMC_CMDR) 1157
23.10.5 SDMMC command response register (SDMMC_RESPCMDR) 1159
23.10.6 SDMMC response x register (SDMMC_RESPxR) 1160
23.10.7 SDMMC data timer register (SDMMC_DTIMER) 1160
23.10.8 SDMMC data length register (SDMMC_DLENR) 1161
23.10.9 SDMMC data control register (SDMMC_DCTRL) 1162
23.10.10 SDMMC data counter register (SDMMC_DCNTR) 1163
23.10.11 SDMMC status register (SDMMC_STAR) 1164
23.10.12 SDMMC interrupt clear register (SDMMC_ICR) 1167
23.10.13 SDMMC mask register (SDMMC_MASKR) 1169
23.10.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER) . . . 1172
23.10.15 SDMMC DMA control register (SDMMC_IDMACTRLR) 1172
23.10.16 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER) 1173
23.10.17 SDMMC IDMA buffer base address register (SDMMC_IDMABASER) 1174
23.10.18 SDMMC IDMA linked list address register (SDMMC_IDMALAR) . . . 1174
23.10.19 SDMMC IDMA linked list memory base register (SDMMC_IDMABAR) 1175
23.10.20 SDMMC data FIFO registers x (SDMMC_FIFORx) 1176
23.10.21 SDMMC register map 1176

24 Delay block (DLYB) 1179

24.1 DLYB introduction 1179

24.2 DLYB main features 1179

24.3 DLYB implementation 1179

24.4 DLYB functional description 1179

24.4.1 DLYB diagram 1179

24.4.2 DLYB pins and internal signals 1180

24.4.3 General description 1180

24.4.4 Delay line length configuration procedure 1181

24.4.5 Output clock phase configuration procedure 1182

24.5 DLYB registers 1183

24.5.1 DLYB control register (DLYB_CR) 1183

24.5.2 DLYB configuration register (DLYB_CFGR) 1183

24.5.3 DLYB register map 1184

25 Analog-to-digital converters (ADC) 1185

25.1 ADC introduction 1185

25.2 ADC main features 1186

25.3 ADC implementation 1187

25.4 ADC functional description 1188

25.4.1 ADC block diagram 1188

25.4.2 ADC pins and internal signals 1189

25.4.3 ADC clocks 1194

25.4.4 ADC connectivity 1196

25.4.5 Slave AHB interface 1200

25.4.6 ADC Deep-power-down mode (DEEPPWD) 1200

25.4.7 Single-ended and differential input channels 1200

25.4.8 Pseudodifferential mode 1201

25.4.9 Calibration (ADCAL, ADCALDIF, ADC_CALFACT) 1201

25.4.10 ADC on-off control (ADEN, ADDIS, ADRDY) 1204

25.4.11 Constraints when writing the ADC control bits 1205

25.4.12 Channel selection (ADC_SQRy, ADC_JSQR) 1206

25.4.13 Channel preselection register (ADC_PCSEL) 1207

25.4.14 Channel-wise programmable sampling time (SMPR1, SMPR2) 1207

25.4.15 Single conversion mode 1209

25.4.16 Continuous conversion mode (CONT = 1) 1210

25.4.17 Discontinuous mode (DISCEN, DISCNUM, JDISCEN) 1210
25.4.18 Starting conversions (ADSTART, JADSTART) 1212
25.4.19 Timing 1213
25.4.20 Stopping an ongoing conversion (ADSTP, JADSTP) 1213
25.4.21 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN) 1215
25.4.22 Injected channel management 1216
25.4.23 Queue of context for injected conversions 1217
25.4.24 Programmable resolution (RES) - fast conversion mode 1225
25.4.25 End of conversion and end of sampling phase (EOC, JEOC, EOSMP) 1226
25.4.26 End of conversion sequence (EOS, JEOS) 1226
25.4.27 Timing diagram examples (single/continuous modes, hardware/software triggers) 1227
25.4.28 Data management 1229
25.4.29 Managing conversions using the ADF 1237
25.4.30 Dynamic low-power features 1238
25.4.31 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT) 1241
25.4.32 Oversampler 1246
25.4.33 Parallel injected oversampling mode 1252
25.4.34 Dual ADC modes 1254
25.4.35 Temperature sensor 1270
25.4.36 VBAT supply monitoring 1272
25.4.37 Monitoring the internal voltage reference 1273
25.4.38 Monitoring the DAC output voltage 1274
25.4.39 Monitoring the supply voltage 1274
25.5 ADC interrupts 1275
25.6 ADC registers (for each ADC) 1276
25.6.1 ADC interrupt and status register (ADC_ISR) 1276
25.6.2 ADC interrupt enable register (ADC_IER) 1279
25.6.3 ADC control register (ADC_CR) 1281
25.6.4 ADC configuration register (ADC_CFGR1) 1284
25.6.5 ADC configuration register 2 (ADC_CFGR2) 1287
25.6.6 ADC sample time register 1 (ADC_SMPR1) 1290
25.6.7 ADC sample time register 2 (ADC_SMPR2) 1291
25.6.8 ADC channel preselection register (ADC_PCSEL) 1291

25.6.9 ADC regular sequence register 1 (ADC_SQR1) 1292
25.6.10 ADC regular sequence register 2 (ADC_SQR2) 1293
25.6.11 ADC regular sequence register 3 (ADC_SQR3) 1294
25.6.12 ADC regular sequence register 4 (ADC_SQR4) 1295
25.6.13 ADC regular data register (ADC_DR) 1296
25.6.14 ADC injected sequence register (ADC_JSQR) 1296
25.6.15 ADC offset y configuration register (ADC_OFCFGRy) 1298
25.6.16 ADC offset y register (ADC_OFRy) 1299
25.6.17 ADC gain compensation register (ADC_GCOMP) 1299
25.6.18 ADC analog watchdog y source channel register (ADC_AWDySR) . 1300
25.6.19 ADC injected channel y data register (ADC_JDRy) 1301
25.6.20 ADC configuration register 3 (ADC_CFGR3) 1301
25.6.21 ADC injected oversampling counter register 1 (ADC_JOVSCR1) . . . 1303
25.6.22 ADC injected oversampling counter register 2 (ADC_JOVSCR2) . . . 1304
25.6.23 ADC analog watchdog 2 configuration register (ADC_AWD2CR) . . . 1305
25.6.24 ADC analog watchdog 3 configuration register (ADC_AWD3CR) . . . 1305
25.6.25 ADC analog watchdog 1 lower threshold register (ADC_AWD1LTR) 1306
25.6.26 ADC analog watchdog 1 higher threshold register (ADC_AWD1HTR) 1306
25.6.27 ADC analog watchdog 2 lower threshold register (ADC_AWD2LTR) 1307
25.6.28 ADC analog watchdog 2 higher threshold register (ADC_AWD2HTR) 1307
25.6.29 ADC analog watchdog 3 lower threshold register (ADC_AWD3LTR) 1308
25.6.30 ADC analog watchdog 3 higher threshold register (ADC_AWD3HTR) 1308
25.6.31 ADC differential mode selection register (ADC_DIFSEL) 1309
25.6.32 ADC calibration factors (ADC_CALFACT) 1309
25.6.33 ADC option register (ADC_OR) 1311
25.7 ADC common registers 1313
25.7.1 ADC common status register (ADCC_CSR) 1313
25.7.2 ADC common control register (ADCC_CCR) 1314
25.7.3 ADC common regular data register for dual mode (ADCC_CDR) 1316
25.7.4 ADC common regular data register for dual mode (ADCC_CDR2) 1317
25.8 ADC register map 1317

26 Digital temperature sensor (DTS) 1322

26.1 DTS introduction 1322
26.2 DTS main features 1322
26.3 DTS functional description 1323
26.3.1 DTS block diagram 1323
26.3.2 DTS internal signals 1323
26.3.3 DTS block operation 1324
26.3.4 Operating modes 1324
26.3.5 Calibration 1324
26.3.6 Prescaler 1324
26.3.7 Temperature measurement principles 1325
26.3.8 Sampling time 1326
26.3.9 Quick measurement mode 1326
26.3.10 Trigger input 1327
26.3.11 On-off control and ready flag 1327
26.3.12 Temperature measurement sequence 1328
26.4 DTS low-power modes 1329
26.5 DTS interrupts 1329
26.5.1 Temperature window comparator 1329
26.5.2 Synchronous interrupt 1329
26.5.3 Asynchronous wake-up 1329
26.6 DTS registers 1330
26.6.1 Temperature sensor configuration register 1 (DTS_CFGR1) 1330
26.6.2 Temperature sensor T0 value register 1 (DTS_T0VALR1) 1331
26.6.3 Temperature sensor ramp value register (DTS_RAMPVALR) 1332
26.6.4 Temperature sensor interrupt threshold register 1 (DTS_ITR1) 1332
26.6.5 Temperature sensor data register (DTS_DR) 1333
26.6.6 Temperature sensor status register (DTS_SR) 1333
26.6.7 Temperature sensor interrupt enable register (DTS_ITENR) 1334
26.6.8 Temperature sensor clear interrupt flag register (DTS_ICIFR) 1336
26.6.9 Temperature sensor option register (DTS_OR) 1336
26.6.10 DTS register map 1337

27 Digital-to-analog converter (DAC) 1339
27.1 DAC introduction 1339
27.2 DAC main features 1339
27.3 DAC implementation 1340

27.4 DAC functional description 1341
27.4.1 DAC block diagram 1341
27.4.2 DAC pins and internal signals 1342
27.4.3 DAC channel enable 1344
27.4.4 DAC data format 1345
27.4.5 DAC conversion 1346
27.4.6 DAC output voltage 1347
27.4.7 DAC trigger selection 1348
27.4.8 DMA requests 1348
27.4.9 Noise generation 1349
27.4.10 Triangle-wave generation 1350
27.4.11 DAC sawtooth wave generation 1351
27.4.12 DAC extended sawtooth wave generation 1353
27.4.13 DAC channel modes 1356
27.4.14 DAC channel buffer calibration 1359
27.4.15 DAC channel conversion modes 1360
27.4.16 Dual DAC channel conversion modes (if dual channels are available) 1361
27.5 DAC in low-power modes 1366
27.6 DAC interrupts 1367
27.7 DAC registers 1367
27.7.1 DAC control register (DAC_CR) 1367
27.7.2 DAC software trigger register (DAC_SWTRGR) 1370
27.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1) 1371
27.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1) 1372
27.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1) 1372
27.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2) 1373
27.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2) 1373
27.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2) 1374
27.7.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD) 1374
27.7.10 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD) 1375

27.7.11 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD) . . . 1375
27.7.12 DAC channel1 data output register (DAC_DOR1) . . . 1376
27.7.13 DAC channel2 data output register (DAC_DOR2) . . . 1376
27.7.14 DAC status register (DAC_SR) . . . 1377
27.7.15 DAC calibration control register (DAC_CCR) . . . 1378
27.7.16 DAC mode control register (DAC_MCR) . . . 1379
27.7.17 DAC channel1 sample and hold sample time register (DAC_SHSR1) . . . 1380
27.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2) . . . 1381
27.7.19 DAC sample and hold time register (DAC_SHHR) . . . 1381
27.7.20 DAC sample and hold refresh time register (DAC_SHRR) . . . 1382
27.7.21 DAC channel1 sawtooth register (DAC_STR1) . . . 1383
27.7.22 DAC channel2 sawtooth register (DAC_STR2) . . . 1383
27.7.23 DAC sawtooth mode register (DAC_STMODR) . . . 1384
27.7.24 DAC extended sawtooth reset control register (DAC_STEXTRST) . 1385
27.7.25 DAC extended sawtooth minimum and maximum value register (DAC_STEXTMM1) . . . 1385
27.7.26 DAC extended sawtooth minimum and maximum value register (DAC_STEXTMM2) . . . 1386
27.7.27 DAC option register (DAC_OR) . . . 1386
27.7.28 DAC register map . . . 1387

28 Voltage reference buffer (VREFBUF) . . . 1390

28.1 VREFBUF introduction . . . 1390
28.2 VREFBUF implementation . . . 1390
28.3 VREFBUF functional description . . . 1390
28.4 VREFBUF trimming . . . 1392
28.5 VREFBUF registers . . . 1393
28.5.1 VREFBUF control and status register (VREFBUF_CSR) . . . 1393
28.5.2 VREFBUF calibration control register (VREFBUF_CCR) . . . 1394
28.5.3 VREFBUF register map . . . 1395

29 Comparator (COMP) . . . 1396

29.1 COMP introduction . . . 1396
29.2 COMP main features . . . 1396
29.3 Implementation . . . 1397

29.4 COMP functional description 1398
29.4.1 COMP block diagram 1398
29.4.2 COMP pins and internal signals 1398
29.4.3 COMP startup stabilization time 1401
29.4.4 COMP power and speed modes 1401
29.4.5 COMP reset and clocks 1402
29.4.6 COMP LOCK mechanism 1402
29.4.7 COMP window mode 1402
29.4.8 COMP hysteresis 1403
29.4.9 COMP deglitcher 1403
29.4.10 COMP digital signal filter 1403
29.4.11 COMP output blanking function 1404
29.4.12 COMP option register 1405
29.5 COMP low-power modes 1405
29.6 COMP interrupts 1406
29.6.1 Interrupt through the EXTI block 1406
29.6.2 Direct interrupt 1407
29.7 SCALER function 1408
29.8 COMP registers 1409
29.8.1 Comparator status register (COMP_SR) 1409
29.8.2 Comparator interrupt clear flag register (COMP_ICFR) 1409
29.8.3 Comparator configuration register (COMP_CFGR) 1410
29.8.4 Comparator filter configuration register (COMP_FLTR) 1411
29.8.5 Comparator option register (COMP_OR) 1412
29.8.6 COMP register map 1413

30 Audio digital filter (ADF) 1414
30.1 ADF introduction 1414
30.2 ADF main features 1414
30.3 ADF implementation 1415
30.4 ADF functional description 1416
30.4.1 ADF block diagram 1416
30.4.2 ADF pins and internal signals 1416
30.4.3 Serial input interface (SITF) 1417
30.4.4 ADC slave interface (ADCITF) 1422
30.4.5 Clock generator (CKGEN) 1422

30.4.6 Bitstream matrix (BSMX) . . . 1424
30.4.7 Digital filter processing (DFLT) . . . 1425
30.4.8 Digital filter acquisition modes . . . 1435
30.4.9 Start-up sequence examples . . . 1443
30.4.10 Sound activity detection (SAD) . . . 1444
30.4.11 Data transfer to memory . . . 1452
30.4.12 Autonomous mode . . . 1455
30.4.13 Register protection . . . 1455
30.5 ADF low-power modes . . . 1456
30.6 ADF interrupts . . . 1456
30.7 ADF application information . . . 1458
30.7.1 ADF configuration examples for audio capture . . . 1458
30.7.2 Programming examples . . . 1459
30.7.3 Connection examples . . . 1461
30.7.4 Global frequency response . . . 1462
30.7.5 Total ADF gain . . . 1463
30.7.6 How to compute SAD thresholds . . . 1466
30.8 ADF registers . . . 1470
30.8.1 ADF global control register (ADF_GCR) . . . 1470
30.8.2 ADF clock generator control register (ADF_CKGCR) . . . 1470
30.8.3 ADF serial interface control register 0 (ADF_SITF0CR) . . . 1472
30.8.4 ADF bitstream matrix control register 0 (ADF_BSMX0CR) . . . 1474
30.8.5 ADF digital filter control register 0 (ADF_DFLT0CR) . . . 1474
30.8.6 ADF digital filer configuration register 0 (ADF_DFLT0CICR) . . . 1476
30.8.7 ADF reshape filter configuration register 0 (ADF_DFLT0RSFR) . . . 1477
30.8.8 ADF delay control register 0 (ADF_DLY0CR) . . . 1478
30.8.9 ADF DFLT0 interrupt enable register (ADF_DFLT0IER) . . . 1479
30.8.10 ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR) . . . 1480
30.8.11 ADF SAD control register (ADF_SADCR) . . . 1481
30.8.12 ADF SAD configuration register (ADF_SADCFGR) . . . 1483
30.8.13 ADF SAD sound level register (ADF_SADSDLVR) . . . 1484
30.8.14 ADF SAD ambient noise level register (ADF_SADANLVR) . . . 1485
30.8.15 ADF digital filter data register 0 (ADF_DFLT0DR) . . . 1485
30.8.16 ADF register map . . . 1485

31 True random number generator (RNG) . . . 1488

31.1 RNG introduction . . . 1488

31.2 RNG main features . . . 1488
31.3 RNG functional description . . . 1489
31.3.1 RNG block diagram . . . 1489
31.3.2 RNG internal signals . . . 1489
31.3.3 Random number generation . . . 1490
31.3.4 RNG initialization . . . 1492
31.3.5 RNG operation . . . 1493
31.3.6 RNG clocking . . . 1495
31.3.7 Error management . . . 1495
31.3.8 RNG low-power use . . . 1497
31.4 RNG interrupts . . . 1497
31.5 RNG processing time . . . 1498
31.6 RNG entropy source validation . . . 1498
31.6.1 Introduction . . . 1498
31.6.2 Validation conditions . . . 1498
31.7 RNG registers . . . 1500
31.7.1 RNG control register (RNG_CR) . . . 1500
31.7.2 RNG status register (RNG_SR) . . . 1502
31.7.3 RNG data register (RNG_DR) . . . 1503
31.7.4 RNG noise source control register (RNG_NSCR) . . . 1504
31.7.5 RNG health test control register x (RNG_HTCRx) . . . 1504
31.7.6 RNG health test status register 0 (RNG_HTSR0) . . . 1505
31.7.7 RNG health test status register 1 (RNG_HTSR1) . . . 1505
31.7.8 RNG noise source mask register (RNG_NSMR) . . . 1506
31.7.9 RNG register map . . . 1507

32 Hash processor (HASH) . . . 1508
32.1 HASH introduction . . . 1508
32.2 HASH main features . . . 1508
32.3 HASH implementation . . . 1509
32.4 HASH functional description . . . 1509
32.4.1 HASH block diagram . . . 1509
32.4.2 HASH internal signals . . . 1509
32.4.3 About secure hash algorithms . . . 1510
32.4.4 Message data feeding . . . 1510
32.4.5 Message digest computing . . . 1511

32.4.6 Message padding . . . 1513
32.4.7 HMAC operation . . . 1515
32.4.8 HASH suspend/resume operations . . . 1516
32.4.9 HASH DMA interface . . . 1518
32.4.10 HASH error management . . . 1519
32.4.11 HASH processing time . . . 1519
32.5 HASH interrupts . . . 1520
32.6 HASH registers . . . 1520
32.6.1 HASH control register (HASH_CR) . . . 1520
32.6.2 HASH data input register (HASH_DIN) . . . 1522
32.6.3 HASH start register (HASH_STR) . . . 1523
32.6.4 HASH digest registers . . . 1524
32.6.5 HASH interrupt enable register (HASH_IMR) . . . 1526
32.6.6 HASH status register (HASH_SR) . . . 1526
32.6.7 HASH context swap registers . . . 1527
32.6.8 HASH register map . . . 1528

33 High-resolution timer (HRTIM) . . . 1530
33.1 Introduction . . . 1530
33.2 Main features . . . 1531
33.3 Functional description . . . 1532
33.3.1 General description . . . 1532
33.3.2 HRTIM pins and internal signals . . . 1534
33.3.3 Clocks . . . 1540
33.3.4 Timer A..F timing units . . . 1543
33.3.5 Master timer . . . 1570
33.3.6 Up-down counting mode . . . 1572
33.3.7 Set/reset events priorities and narrow pulses management . . . 1579
33.3.8 External events global conditioning . . . 1580
33.3.9 External event filtering in timing units . . . 1585
33.3.10 Delayed protection . . . 1592
33.3.11 Register preload and update management . . . 1598
33.3.12 PWM mode with “greater than” comparison . . . 1602
33.3.13 Events propagation within or across multiple timers . . . 1603
33.3.14 Output management . . . 1608
33.3.15 Burst mode controller . . . 1610
33.3.16 Chopper . . . 1618

33.3.17 Fault protection . . . 1619
33.3.18 Auxiliary outputs . . . 1624
33.3.19 Synchronizing the HRTIM with other timers or HRTIM instances . . . 1627
33.3.20 ADC triggers . . . 1630
33.3.21 DAC triggers . . . 1634
33.3.22 Interrupts . . . 1638
33.3.23 DMA . . . 1640
33.3.24 HRTIM initialization . . . 1644
33.3.25 Debug . . . 1645
33.4 Application use cases . . . 1646
33.4.1 Buck converter . . . 1646
33.4.2 Buck converter with synchronous rectification . . . 1647
33.4.3 Multiphase converters . . . 1648
33.4.4 Transition mode power factor correction . . . 1649
33.5 HRTIM registers . . . 1652
33.5.1 HRTIM master timer control register (HRTIM_MCR) . . . 1652
33.5.2 HRTIM master timer interrupt status register (HRTIM_MISR) . . . 1655
33.5.3 HRTIM master timer interrupt clear register (HRTIM_MICR) . . . 1656
33.5.4 HRTIM master timer DMA interrupt enable register (HRTIM_MDIER) . . . 1657
33.5.5 HRTIM master timer counter register (HRTIM_MCNTR) . . . 1659
33.5.6 HRTIM master timer period register (HRTIM_MPER) . . . 1659
33.5.7 HRTIM master timer repetition register (HRTIM_MREP) . . . 1660
33.5.8 HRTIM master timer compare 1 register (HRTIM_MCMP1R) . . . 1660
33.5.9 HRTIM master timer compare 2 register (HRTIM_MCMP2R) . . . 1661
33.5.10 HRTIM master timer compare 3 register (HRTIM_MCMP3R) . . . 1661
33.5.11 HRTIM master timer compare 4 register (HRTIM_MCMP4R) . . . 1662
33.5.12 HRTIM timer x control register (HRTIM_TIMxCR) (x = A to F) . . . 1663
33.5.13 HRTIM timer x interrupt status register (HRTIM_TIMxISR) (x = A to F) . . . 1667
33.5.14 HRTIM timer x interrupt clear register (HRTIM_TIMxICR) (x = A to F) . . . 1669
33.5.15 HRTIM timer x DMA interrupt enable register (HRTIM_TIMxDIER) (x = A to F) . . . 1671
33.5.16 HRTIM timer x counter register (HRTIM_CNTxR) (x = A to F) . . . 1674
33.5.17 HRTIM timer x period register (HRTIM_PERxR) (x = A to F) . . . 1674
33.5.18 HRTIM timer x repetition register (HRTIM_REPxR) (x = A to F) . . . 1676
33.5.19 HRTIM timer x compare 1 register (HRTIM_CMP1xR) (x = A to F) . . 1676

33.5.20 HRTIM timer x compare 1 compound register (HRTIM_CMP1CxR) (x = A to F) 1678
33.5.21 HRTIM timer x compare 2 register (HRTIM_CMP2xR) (x = A to F) . . 1678
33.5.22 HRTIM timer x compare 3 register (HRTIM_CMP3xR) (x = A to F) . . 1680
33.5.23 HRTIM timer x compare 4 register (HRTIM_CMP4xR) (x = A to F) . . 1681
33.5.24 HRTIM timer x capture 1 register (HRTIM_CPT1xR) (x = A to F) . . . 1682
33.5.25 HRTIM timer x capture 2 register (HRTIM_CPT2xR) (x = A to F) 1683
33.5.26 HRTIM timer x deadtime register (HRTIM_DTxR) (x = A to F) 1684
33.5.27 HRTIM timer x output 1 set register (HRTIM_SETx1R) (x = A to F) . 1686
33.5.28 HRTIM timer x output 1 reset register (HRTIM_RSTx1R) (x = A to F) 1689
33.5.29 HRTIM timer x output 2 set register (HRTIM_SETx2R) (x = A to F) . 1691
33.5.30 HRTIM timer x output 2 reset register (HRTIM_RSTx2R) (x = A to F) 1694
33.5.31 HRTIM timer x external event filtering register 1 (HRTIM_EEFxR1) (x = A to F) 1697
33.5.32 HRTIM timer x external event filtering register 2 (HRTIM_EEFxR2) (x = A to F) 1700
33.5.33 HRTIM timer A reset register (HRTIM_RSTAR) 1701
33.5.34 HRTIM timer B reset register (HRTIM_RSTBR) 1703
33.5.35 HRTIM timer C reset register (HRTIM_RSTCR) 1705
33.5.36 HRTIM timer D reset register (HRTIM_RSTDR) 1707
33.5.37 HRTIM timer E reset register (HRTIM_RSTER) 1709
33.5.38 HRTIM timer F reset register (HRTIM_RSTFR) 1711
33.5.39 HRTIM timer x chopper register (HRTIM_CHPxR) (x = A to F) 1713
33.5.40 HRTIM timer A capture 1 control register (HRTIM_CPT1ACR) 1714
33.5.41 HRTIM timer B capture 1 control register (HRTIM_CPT1BCR) 1716
33.5.42 HRTIM timer C capture 1 control register (HRTIM_CPT1CCR) 1718
33.5.43 HRTIM timer D capture 1 control register (HRTIM_CPT1DCR) 1720
33.5.44 HRTIM timer E capture 1 control register (HRTIM_CPT1ECR) 1722
33.5.45 HRTIM timer F capture 1 control register (HRTIM_CPT1FCR) 1724
33.5.46 HRTIM timer A capture 2 control register (HRTIM_CPT2ACR) 1726
33.5.47 HRTIM timer B capture 2 control register (HRTIM_CPT2BCR) 1728
33.5.48 HRTIM timer C capture 2 control register (HRTIM_CPT2CCR) 1730
33.5.49 HRTIM timer D capture 2 control register (HRTIM_CPT2DCR) 1732
33.5.50 HRTIM timer E capture 2 control register (HRTIM_CPT2ECR) 1734
33.5.51 HRTIM timer F capture 2 control register (HRTIM_CPT2FCR) 1736
33.5.52 HRTIM timer x output register (HRTIM_OUTxR) (x = A to F) 1738
33.5.53 HRTIM timer x fault register (HRTIM_FLTxR) (x = A to F) 1741

33.5.54 HRTIM timer x control register 2 (HRTIM_TIMxCR2) (x = A to F) . . . 1743
33.5.55 HRTIM timer x external event filtering register 3 (HRTIM_TIMxEEFR3) (x = A to F) . . . 1747
33.5.56 HRTIM timer x external event selection register (HRTIM_EESELxR) (x = A to F) . . . 1749
33.5.57 HRTIM timer x crossbar override trigger register (HRTIM_COTRGxR) (x = A to F) . . . 1752
33.5.58 HRTIM timer x crossbar override forcing event register (HRTIM_COFExR) (x = A to F) . . . 1755
33.5.59 HRTIM timer x crossbar override delayed fault event register (HRTIM_CODExR) (x = A to F) . . . 1758
33.5.60 HRTIM control register 1 (HRTIM_CR1) . . . 1759
33.5.61 HRTIM control register 2 (HRTIM_CR2) . . . 1761
33.5.62 HRTIM interrupt status register (HRTIM_ISR) . . . 1763
33.5.63 HRTIM interrupt clear register (HRTIM_ICR) . . . 1764
33.5.64 HRTIM interrupt enable register (HRTIM_IER) . . . 1765
33.5.65 HRTIM output enable register (HRTIM_OENR) . . . 1766
33.5.66 HRTIM output disable register (HRTIM_ODISR) . . . 1767
33.5.67 HRTIM output disable status register (HRTIM_ODSR) . . . 1768
33.5.68 HRTIM burst mode control register (HRTIM_BMCR) . . . 1769
33.5.69 HRTIM burst mode trigger register (HRTIM_BMTRGR) . . . 1771
33.5.70 HRTIM burst mode compare register (HRTIM_BMCMPR) . . . 1773
33.5.71 HRTIM burst mode period register (HRTIM_BMPER) . . . 1773
33.5.72 HRTIM timer external event control register 1 (HRTIM_EECR1) . . . 1774
33.5.73 HRTIM timer external event control register 2 (HRTIM_EECR2) . . . 1776
33.5.74 HRTIM timer external event control register 3 (HRTIM_EECR3) . . . 1777
33.5.75 HRTIM ADC trigger 1 register (HRTIM_ADC1R) . . . 1778
33.5.76 HRTIM ADC trigger 2 register (HRTIM_ADC2R) . . . 1781
33.5.77 HRTIM ADC trigger 3 register (HRTIM_ADC3R) . . . 1783
33.5.78 HRTIM ADC trigger 4 register (HRTIM_ADC4R) . . . 1785
33.5.79 HRTIM fault input register 1 (HRTIM_FLTINR1) . . . 1787
33.5.80 HRTIM fault input register 2 (HRTIM_FLTINR2) . . . 1789
33.5.81 HRTIM burst DMA master timer update register (HRTIM_BDMUPR) 1791
33.5.82 HRTIM burst DMA timer x update register (HRTIM_BDTxUPR) (x = A to F) . . . 1792
33.5.83 HRTIM burst DMA data register (HRTIM_BDMADR) . . . 1794
33.5.84 HRTIM ADC extended trigger register (HRTIM_ADCER) . . . 1795
33.5.85 HRTIM ADC trigger update register (HRTIM_ADCUR) . . . 1798
33.5.86 HRTIM ADC post scaler register 1 (HRTIM_ADCPS1) . . . 1799

33.5.87 HRTIM ADC post scaler register 2 (HRTIM_ADCPS2) 1800
33.5.88 HRTIM fault input register 3 (HRTIM_FLTINR3) 1801
33.5.89 HRTIM fault input register 4 (HRTIM_FLTINR4) 1803
33.5.90 HRTIM fault input register 5 (HRTIM_FLTINR5) 1805
33.5.91 HRTIM timer external event control register 4 (HRTIM_EECR4) . . . 1807
33.5.92 HRTIM timer external event control register 5 (HRTIM_EECR5) . . . 1809
33.5.93 HRTIM timer external event control register 6 (HRTIM_EECR6) . . . 1811
33.5.94 HRTIM timer external event control register 7 (HRTIM_EECR7) . . . 1812
33.5.95 HRTIM control register 3 (HRTIM_CR3) 1814
33.5.96 HRTIM option register (HRTIM_OR) 1814
33.5.97 HRTIM register map 1816

34 Advanced-control timers (TIM1/TIM8/TIM20) 1827

34.1 TIM1/TIM8/TIM20 introduction 1827
34.2 TIM1/TIM8/TIM20 main features 1827
34.3 TIM1/TIM8/TIM20 functional description 1828
34.3.1 Block diagram 1828
34.3.2 TIM1/TIM8/TIM20 pins and internal signals 1829
34.3.3 Time-base unit 1835
34.3.4 Counter modes 1837
34.3.5 Repetition counter 1849
34.3.6 External trigger input 1850
34.3.7 Clock selection 1851
34.3.8 Capture/compare channels 1855
34.3.9 Input capture mode 1858
34.3.10 PWM input mode 1859
34.3.11 Forced output mode 1860
34.3.12 Output compare mode 1860
34.3.13 PWM mode 1861
34.3.14 Asymmetric PWM mode 1870
34.3.15 Combined PWM mode 1873
34.3.16 Combined 3-phase PWM mode 1876
34.3.17 Complementary outputs and dead-time insertion 1877
34.3.18 Clock dividers for dead-time generators and digital filters 1880
34.3.19 Using the break function 1880
34.3.20 Delayed break 1886
34.3.21 Bidirectional break inputs 1888

34.3.22 Output idle state override 1890
34.3.23 Clearing the tim_ocxref signal on an external event 1891
34.3.24 6-step PWM generation 1894
34.3.25 One-pulse mode 1895
34.3.26 Retriggerable One-pulse mode 1897
34.3.27 Pulse on compare mode 1898
34.3.28 Encoder interface mode 1900
34.3.29 Encoder with built-in debouncer 1917
34.3.30 Direction bit output 1919
34.3.31 Clock drift measurement 1919
34.3.32 UIF bit remapping 1920
34.3.33 Timer input XOR function 1920
34.3.34 Interfacing with Hall sensors 1922
34.3.35 Timer synchronization 1923
34.3.36 ADC triggers 1928
34.3.37 ADC synchronization 1928
34.3.38 DMA burst mode 1929
34.3.39 TIM1/TIM8/TIM20 DMA requests 1930
34.3.40 Debug mode 1931
34.4 TIM1/TIM8/TIM20 low-power modes 1931
34.5 TIM1/TIM8/TIM20 interrupts 1931
34.6 TIM1/TIM8/TIM20 registers 1933
34.6.1 TIM control register 1 (TIM_CR1) 1933
34.6.2 TIM control register 2 (TIM_CR2) 1935
34.6.3 TIM slave mode control register (TIM_SMCR) 1940
34.6.4 TIM DMA/interrupt enable register (TIM_DIER) 1944
34.6.5 TIM status register (TIM_SR) 1945
34.6.6 TIM event generation register (TIM_EGR) 1949
34.6.7 TIM capture/compare mode register 1 (TIM_CCMR1) 1950
34.6.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1) . . . 1952
34.6.9 TIM capture/compare mode register 2 (TIM_CCMR2) 1956
34.6.10 TIM capture/compare mode register 2 [alternate] (TIM_CCMR2) . . . 1958
34.6.11 TIM capture/compare enable register (TIM_CCER) 1961
34.6.12 TIM counter (TIM_CNT) 1965
34.6.13 TIM prescaler (TIM_PSC) 1965
34.6.14 TIM autoreload register (TIM_ARR) 1966
34.6.15 TIM repetition counter register (TIM_RCR) 1966

34.6.16 TIM capture/compare register 1 (TIM_CCR1) 1967
34.6.17 TIM capture/compare register 2 (TIM_CCR2) 1967
34.6.18 TIM capture/compare register 3 (TIM_CCR3) 1968
34.6.19 TIM capture/compare register 4 (TIM_CCR4) 1969
34.6.20 TIM break and dead-time register (TIM_BDTR) 1970
34.6.21 TIM capture/compare register 5 (TIM_CCR5) 1974
34.6.22 TIM capture/compare register 6 (TIM_CCR6) 1976
34.6.23 TIM capture/compare mode register 3 (TIM_CCMR3) 1977
34.6.24 TIM timer deadtime register 2 (TIM_DTR2) 1980
34.6.25 TIM timer encoder control register (TIM_ECR) 1981
34.6.26 TIM timer input selection register (TIM_TISEL) 1982
34.6.27 TIM alternate function option register 1 (TIM_AF1) 1983
34.6.28 TIM alternate function register 2 (TIM_AF2) 1988
34.6.29 TIM capture/compare register 7 (TIM_CCR7) 1992
34.6.30 TIM capture/compare mode register 4 (TIM_CCMR4) 1993
34.6.31 TIM multilevel protection register 1 (TIM_MPR1) 1995
34.6.32 TIM multilevel protection register 2 (TIM_MPR2) 1997
34.6.33 TIM output override register (TIM_OOR) 1999
34.6.34 TIM DMA control register (TIM_DCR) 2000
34.6.35 TIM DMA address for full transfer (TIM_DMAR) 2002
34.6.36 TIM1/TIM8/TIM20 register map 2002

35 General-purpose timers (TIM2/TIM3/TIM4/TIM5) 2006

35.1 TIM2/TIM3/TIM4/TIM5 introduction 2006
35.2 TIM2/TIM3/TIM4/TIM5 main features 2006
35.3 TIM2/TIM3/TIM4/TIM5 implementation 2007
35.4 TIM2/TIM3/TIM4/TIM5 functional description 2008
35.4.1 Block diagram 2008
35.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals 2009
35.4.3 Time-base unit 2014
35.4.4 Counter modes 2016
35.4.5 External trigger input 2028
35.4.6 Clock selection 2029
35.4.7 Capture/compare channels 2033
35.4.8 Input capture mode 2035
35.4.9 PWM input mode 2036
35.4.10 Forced output mode 2037

35.4.11 Output compare mode 2037
35.4.12 PWM mode 2039
35.4.13 Asymmetric PWM mode 2047
35.4.14 Combined PWM mode 2049
35.4.15 Clock dividers for dead-time generators and digital filters 2050
35.4.16 Clearing the tim_ocxref signal on an external event 2051
35.4.17 One-pulse mode 2054
35.4.18 Retriggerable one-pulse mode 2056
35.4.19 Pulse on compare mode 2057
35.4.20 Encoder interface mode 2059
35.4.21 Encoder with built-in debouncer 2077
35.4.22 Direction bit output 2079
35.4.23 Clock drift measurement 2079
35.4.24 UIF bit remapping 2080
35.4.25 Timer input XOR function 2080
35.4.26 Timers and external trigger synchronization 2082
35.4.27 Timer synchronization 2085
35.4.28 ADC triggers 2090
35.4.29 ADC synchronization 2092
35.4.30 DMA burst mode 2092
35.4.31 TIM2/TIM3/TIM4/TIM5 DMA requests 2093
35.4.32 Debug mode 2094
35.4.33 TIM2/TIM3/TIM4/TIM5 low-power modes 2094
35.4.34 TIM2/TIM3/TIM4/TIM5 interrupts 2095
35.5 TIM2/TIM3/TIM4/TIM5 registers 2096
35.5.1 TIM control register 1 (TIM_CR1) 2096
35.5.2 TIM control register 2 (TIM_CR2) 2098
35.5.3 TIM slave mode control register (TIM_SMCR) 2101
35.5.4 TIM DMA/interrupt enable register (TIM_DIER) 2105
35.5.5 TIM status register (TIM_SR) 2106
35.5.6 TIM event generation register (TIM_EGR) 2109
35.5.7 TIM capture/compare mode register 1 (TIM_CCMR1) 2110
35.5.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1) . . . 2112
35.5.9 TIM capture/compare mode register 2 (TIM_CCMR2) 2114
35.5.10 TIM capture/compare mode register 2 [alternate] (TIM_CCMR2) . . . 2115
35.5.11 TIM capture/compare enable register (TIM_CCER) 2118
35.5.12 TIM counter (TIM_CNT) 2120

35.5.13 TIM prescaler (TIM_PSC) 2120
35.5.14 TIM autoreload register (TIM_ARR) 2121
35.5.15 TIM capture/compare register 1 (TIM_CCR1) 2121
35.5.16 TIM capture/compare register 2 (TIM_CCR2) 2122
35.5.17 TIM capture/compare register 3 (TIM_CCR3) 2123
35.5.18 TIM capture/compare register 4 (TIM_CCR4) 2124
35.5.19 TIM timer encoder control register (TIM_ECR) 2125
35.5.20 TIM timer input selection register (TIM_TISEL) 2126
35.5.21 TIM alternate function register 1 (TIM_AF1) 2127
35.5.22 TIM alternate function register 2 (TIM_AF2) 2128
35.5.23 TIM DMA control register (TIM_DCR) 2129
35.5.24 TIM DMA address for full transfer (TIM_DMAR) 2130
35.5.25 TIM2/TIM3/TIM4/TIM5 register map 2131

36 Basic timers (TIM6/TIM7) 2134

36.1 TIM6/TIM7 introduction 2134
36.2 TIM6/TIM7 main features 2134
36.3 TIM6/TIM7 functional description 2135
36.3.1 TIM6/TIM7 block diagram 2135
36.3.2 TIM6/TIM7 internal signals 2135
36.3.3 TIM6/TIM7 clocks 2136
36.3.4 Time-base unit 2136
36.3.5 Counting mode 2138
36.3.6 UIF bit remapping 2145
36.3.7 ADC triggers 2146
36.3.8 ADC synchronization 2146
36.3.9 TIM6/TIM7 DMA requests 2147
36.3.10 Debug mode 2147
36.3.11 TIM6/TIM7 low-power modes 2147
36.3.12 TIM6/TIM7 interrupts 2147
36.4 TIM6/TIM7 registers 2148
36.4.1 TIM control register 1 (TIM_CR1) 2148
36.4.2 TIM control register 2 (TIM_CR2) 2150
36.4.3 TIM DMA/interrupt enable register (TIM_DIER) 2150
36.4.4 TIM status register (TIM_SR) 2152
36.4.5 TIM event generation register (TIM_EGR) 2152
36.4.6 TIM counter (TIM_CNT) 2153

36.4.7 TIM prescaler (TIM_PSC) 2153
36.4.8 TIM autoreload register (TIM_ARR) 2154
36.4.9 TIM6/TIM7 register map 2155

37 General-purpose timers (TIM9/TIM10/TIM11/TIM12/TIM13/TIM14) . 2156

37.1 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 introduction 2156
37.2 TIM9/TIM12 main features 2156
37.3 TIM10/TIM11/TIM13/TIM14 main features 2157
37.4 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 functional description 2158
37.4.1 Block diagram 2158
37.4.2 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 pins and internal signals . . 2159
37.4.3 Time-base unit 2162
37.4.4 Counter modes 2164
37.4.5 Clock selection 2167
37.4.6 Capture/compare channels 2169
37.4.7 Input capture mode 2171
37.4.8 PWM input mode (TIM9/TIM12 only) 2172
37.4.9 Forced output mode 2173
37.4.10 Output compare mode 2174
37.4.11 PWM mode 2175
37.4.12 Combined PWM mode (TIM9/TIM12 only) 2180
37.4.13 Clock dividers for digital filters 2181
37.4.14 One-pulse mode 2182
37.4.15 Retriggerable one pulse mode (TIM9/TIM12 only) 2183
37.4.16 UIF bit remapping 2184
37.4.17 Timer input XOR function 2184
37.4.18 TIM9/TIM12 external trigger synchronization 2186
37.4.19 Slave mode – combined reset + trigger mode 2189
37.4.20 Slave mode – combined reset + gated mode 2189
37.4.21 Timer synchronization (TIM9/TIM12 only) 2189
37.4.22 Using timer output as trigger for other timers (TIM10/TIM11/TIM13/TIM14 only) 2189
37.4.23 ADC triggers (TIM9/TIM12 only) 2189
37.4.24 ADC synchronization (TIM9/TIM12 only) 2190
37.4.25 Debug mode 2191
37.5 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 low-power modes 2191
37.6 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 interrupts 2191

37.7 TIM9/TIM12 registers 2191
37.7.1 TIM control register 1 (TIM_CR1) 2192
37.7.2 TIM control register 2 (TIM_CR2) 2193
37.7.3 TIM slave mode control register (TIM_SMCR) 2195
37.7.4 TIM interrupt enable register (TIM_DIER) 2197
37.7.5 TIM status register (TIM_SR) 2198
37.7.6 TIM event generation register (TIM_EGR) 2199
37.7.7 TIM capture/compare mode register 1 (TIM_CCMR1) 2200
37.7.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1) . . . 2201
37.7.9 TIM capture/compare enable register (TIM_CCER) 2204
37.7.10 TIM counter (TIM_CNT) 2205
37.7.11 TIM prescaler (TIM_PSC) 2206
37.7.12 TIM autoreload register (TIM_ARR) 2206
37.7.13 TIM capture/compare register 1 (TIM_CCR1) 2207
37.7.14 TIM capture/compare register 2 (TIM_CCR2) 2207
37.7.15 TIM timer input selection register (TIM_TISEL) 2208
37.7.16 TIM9/TIM12 register map 2209
37.8 TIM10/TIM11/TIM13/TIM14 registers 2211
37.8.1 TIM control register 1 (TIM_CR1) 2211
37.8.2 TIM interrupt enable register (TIM_DIER) 2212
37.8.3 TIM status register (TIM_SR) 2213
37.8.4 TIM event generation register (TIM_EGR) 2214
37.8.5 TIM capture/compare mode register 1 (TIM_CCMR1) 2215
37.8.6 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1) . . . 2216
37.8.7 TIM capture/compare enable register (TIM_CCER) 2217
37.8.8 TIM counter (TIM_CNT) 2218
37.8.9 TIM prescaler (TIM_PSC) 2219
37.8.10 TIM autoreload register (TIM_ARR) 2219
37.8.11 TIM capture/compare register 1 (TIM_CCR1) 2220
37.8.12 TIM timer input selection register (TIM_TISEL) 2220
37.8.13 TIM10/TIM11/TIM13/TIM14 register map 2221

38 General purpose timers (TIM15/TIM16/TIM17) 2223

38.1 TIM15/TIM16/TIM17 introduction 2223
38.2 TIM15 main features 2223
38.3 TIM16/TIM17 main features 2224

38.4 TIM15/TIM16/TIM17 functional description 2225
38.4.1 Block diagram 2225
38.4.2 TIM15/TIM16/TIM17 pins and internal signals 2226
38.4.3 Time-base unit 2231
38.4.4 Counter modes 2233
38.4.5 Repetition counter 2245
38.4.6 Clock selection 2247
38.4.7 Capture/compare channels 2249
38.4.8 Input capture mode 2251
38.4.9 PWM input mode (only for TIM15) 2252
38.4.10 Forced output mode 2253
38.4.11 Output compare mode 2254
38.4.12 PWM mode 2255
38.4.13 Asymmetric PWM mode (TIM15 only) 2262
38.4.14 Combined PWM mode (TIM15 only) 2263
38.4.15 Complementary outputs and dead-time insertion 2264
38.4.16 Clock dividers for dead-time generators and digital filters 2267
38.4.17 Using the break function 2267
38.4.18 Delayed break 2272
38.4.19 Bidirectional break input 2274
38.4.20 Output idle state override 2276
38.4.21 Clearing the tim_ocxref signal on an external event 2276
38.4.22 6-step PWM generation 2278
38.4.23 One-pulse mode 2280
38.4.24 Retriggerable one pulse mode (TIM15 only) 2281
38.4.25 Encoder interface mode 2282
38.4.26 Encoder with built-in debouncer (TIM15 only) 2289
38.4.27 Clock drift measurement 2290
38.4.28 UIF bit remapping 2291
38.4.29 Timer input XOR function (TIM15 only) 2291
38.4.30 External trigger synchronization (TIM15 only) 2293
38.4.31 Slave mode – combined reset + trigger mode (TIM15 only) 2296
38.4.32 Slave mode – combined reset + gated mode (TIM15 only) 2296
38.4.33 Timer synchronization (TIM15 only) 2297
38.4.34 Using timer output as trigger for other timers (TIM16/TIM17 only) . . 2297
38.4.35 ADC triggers (TIM15 only) 2297
38.4.36 ADC synchronization (TIM15 only) 2298

38.4.37 DMA burst mode 2298
38.4.38 TIM15/TIM16/TIM17 DMA requests 2299
38.4.39 Debug mode 2300
38.5 TIM15/TIM16/TIM17 low-power modes 2300
38.6 TIM15/TIM16/TIM17 interrupts 2300
38.7 TIM15 registers 2302
38.7.1 TIM control register 1 (TIM_CR1) 2302
38.7.2 TIM control register 2 (TIM_CR2) 2304
38.7.3 TIM slave mode control register (TIM_SMCR) 2307
38.7.4 TIM DMA/interrupt enable register (TIM_DIER) 2310
38.7.5 TIM status register (TIM_SR) 2311
38.7.6 TIM event generation register (TIM_EGR) 2313
38.7.7 TIM capture/compare mode register 1 (TIM_CCMR1) 2314
38.7.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1) . . . 2316
38.7.9 TIM capture/compare enable register (TIM_CCER) 2319
38.7.10 TIM counter (TIM_CNT) 2322
38.7.11 TIM prescaler (TIM_PSC) 2322
38.7.12 TIM autoreload register (TIM_ARR) 2322
38.7.13 TIM repetition counter register (TIM_RCR) 2323
38.7.14 TIM capture/compare register 1 (TIM_CCR1) 2323
38.7.15 TIM capture/compare register 2 (TIM_CCR2) 2325
38.7.16 TIM break and dead-time register (TIM_BDTR) 2325
38.7.17 TIM timer deadtime register 2 (TIM_DTR2) 2328
38.7.18 TIM input selection register (TIM_TISEL) 2329
38.7.19 TIM alternate function register 1 (TIM_AF1) 2330
38.7.20 TIM alternate function register 2 (TIM_AF2) 2334
38.7.21 TIM multilevel protection register 1 (TIM_MPR1) 2335
38.7.22 TIM multilevel protection register 2 (TIM_MPR2) 2335
38.7.23 TIM output override register (TIM_OOR) 2336
38.7.24 TIM DMA control register (TIM_DCR) 2337
38.7.25 TIM DMA address for full transfer (TIM_DMAR) 2338
38.7.26 TIM15 register map 2339
38.8 TIM16/TIM17 registers 2342
38.8.1 TIM control register 1 (TIM_CR1) 2342
38.8.2 TIM control register 2 (TIM_CR2) 2344
38.8.3 TIM DMA/interrupt enable register (TIM_DIER) 2345
38.8.4 TIM status register (TIM_SR) 2346

38.8.5 TIM event generation register (TIM_EGR) 2347
38.8.6 TIM capture/compare mode register 1 (TIM_CCMR1) 2348
38.8.7 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1) . . . 2349
38.8.8 TIM capture/compare enable register (TIM_CCER) 2351
38.8.9 TIM counter (TIM_CNT) 2354
38.8.10 TIM prescaler (TIM_PSC) 2354
38.8.11 TIM autoreload register (TIM_ARR) 2355
38.8.12 TIM repetition counter register (TIM_RCR) 2355
38.8.13 TIM capture/compare register 1 (TIM_CCR1) 2356
38.8.14 TIM break and dead-time register (TIM_BDTR) 2357
38.8.15 TIM timer deadtime register 2 (TIM_DTR2) 2360
38.8.16 TIM input selection register (TIM_TISEL) 2361
38.8.17 TIM alternate function register 1 (TIM_AF1) 2361
38.8.18 TIM alternate function register 2 (TIM_AF2) 2365
38.8.19 TIM multilevel protection register 1 (TIM_MPR1) 2366
38.8.20 TIM multilevel protection register 2 (TIM_MPR2) 2367
38.8.21 TIM output override register (TIM_OOR) 2367
38.8.22 TIM DMA control register (TIM_DCR) 2368
38.8.23 TIM DMA address for full transfer (TIM_DMAR) 2369
38.8.24 TIM16/TIM17 register map 2370

39 Low-power timer (LPTIM) 2373

39.1 LPTIM introduction 2373
39.2 LPTIM main features 2373
39.3 LPTIM implementation 2374
39.4 LPTIM functional description 2375
39.4.1 LPTIM block diagram 2375
39.4.2 LPTIM pins and internal signals 2376
39.4.3 LPTIM input and trigger mapping 2378
39.4.4 LPTIM reset and clocks 2379
39.4.5 Glitch filter 2380
39.4.6 Prescaler 2381
39.4.7 Trigger multiplexer 2381
39.4.8 Operating mode 2382
39.4.9 Timeout function 2384
39.4.10 Waveform generation 2384
39.4.11 Register update 2385

39.4.12 Counter mode . . . 2386
39.4.13 Timer enable . . . 2386
39.4.14 Timer counter reset . . . 2387
39.4.15 Encoder mode . . . 2387
39.4.16 Repetition counter . . . 2389
39.4.17 Capture/compare channels . . . 2390
39.4.18 Input capture mode . . . 2391
39.4.19 PWM mode . . . 2393
39.4.20 DMA requests . . . 2395
39.4.21 Debug mode . . . 2396
39.5 LPTIM low-power modes . . . 2396
39.6 LPTIM interrupts . . . 2396
39.7 LPTIM registers . . . 2397
39.7.1 LPTIMx interrupt and status register (LPTIMx_ISR)(x = 4 to 5) . . . 2398
39.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3) . . . 2399
39.7.3 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3) . . . 2401
39.7.4 LPTIMx interrupt clear register (LPTIMx_ICR)(x = 4 to 5) . . . 2403
39.7.5 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3) . . . 2404
39.7.6 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3) . . . 2405
39.7.7 LPTIMx interrupt enable register (LPTIMx_DIER)(x = 4 to 5) . . . 2406
39.7.8 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3) . . . 2408
39.7.9 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3) . . . 2409
39.7.10 LPTIM configuration register (LPTIM_CFGR) . . . 2411
39.7.11 LPTIM control register (LPTIM_CR) . . . 2414
39.7.12 LPTIM compare register 1 (LPTIM_CCR1) . . . 2415
39.7.13 LPTIM autoreload register (LPTIM_ARR) . . . 2416
39.7.14 LPTIM counter register (LPTIM_CNT) . . . 2416
39.7.15 LPTIM configuration register 2 (LPTIM_CFGR2) . . . 2417
39.7.16 LPTIM repetition register (LPTIM_RCR) . . . 2418
39.7.17 LPTIM capture/compare mode register 1 (LPTIM_CCMR1) . . . 2418
39.7.18 LPTIM compare register 2 (LPTIM_CCR2) . . . 2421
39.7.19 LPTIM register map . . . 2421

40 Independent watchdog (IWDG) 2424
40.1 IWDG introduction 2424
40.2 IWDG main features 2424
40.3 IWDG implementation 2424
40.4 IWDG functional description 2425
40.4.1 IWDG block diagram 2425
40.4.2 IWDG internal signals 2426
40.4.3 Software and hardware watchdog modes 2426
40.4.4 Window option 2427
40.4.5 Debug 2430
40.4.6 Register access protection 2430
40.5 IWDG low power modes 2431
40.6 IWDG interrupts 2431
40.7 IWDG registers 2433
40.7.1 IWDG key register (IWDG_KR) 2434
40.7.2 IWDG prescaler register (IWDG_PR) 2434
40.7.3 IWDG reload register (IWDG_RLR) 2435
40.7.4 IWDG status register (IWDG_SR) 2435
40.7.5 IWDG window register (IWDG_WINR) 2437
40.7.6 IWDG early wake-up interrupt register (IWDG_EWCR) 2437
40.7.7 IWDG register map 2439

41 System window watchdog (WWDG) 2440
41.1 WWDG introduction 2440
41.2 WWDG main features 2440
41.3 WWDG implementation 2440
41.4 WWDG functional description 2441
41.4.1 WWDG block diagram 2441
41.4.2 WWDG internal signals 2441
41.4.3 Enabling the watchdog 2441
41.4.4 Controlling the down-counter 2442
41.4.5 How to program the watchdog timeout 2442
41.4.6 Debug mode 2443
41.5 WWDG interrupts 2444
41.6 WWDG registers 2444

41.6.1 WWDG control register (WWDG_CR) 2444
41.6.2 WWDG configuration register (WWDG_CFR) 2445
41.6.3 WWDG status register (WWDG_SR) 2445
41.6.4 WWDG register map 2446

42 Real-time clock (RTC) 2447
42.1 RTC introduction 2447
42.2 RTC main features 2447
42.3 RTC functional description 2447
42.3.1 RTC block diagram 2447
42.3.2 RTC pins and internal signals 2449
42.3.3 GPIOs controlled by the RTC 2450
42.3.4 RTC registers write protection 2450
42.3.5 RTC privilege protection modes 2450
42.3.6 Clock and prescalers 2451
42.3.7 Real-time clock and calendar 2453
42.3.8 Calendar ultra-low power mode 2453
42.3.9 Programmable alarms 2453
42.3.10 Periodic auto-wake-up 2454
42.3.11 RTC initialization and configuration 2455
42.3.12 Reading the calendar 2457
42.3.13 Resetting the RTC 2458
42.3.14 RTC synchronization 2459
42.3.15 RTC reference clock detection 2459
42.3.16 RTC smooth digital calibration 2460
42.3.17 Timestamp function 2462
42.3.18 Calibration clock output 2463
42.3.19 Tamper and alarm output 2463
42.4 RTC low-power modes 2464
42.5 RTC interrupts 2464
42.6 RTC registers 2465
42.6.1 RTC time register (RTC_TR) 2465
42.6.2 RTC date register (RTC_DR) 2466
42.6.3 RTC subsecond register (RTC_SSR) 2467
42.6.4 RTC initialization control and status register (RTC_ICSR) 2468
42.6.5 RTC prescaler register (RTC_PRER) 2470

42.6.6 RTC wake-up timer register (RTC_WUTR) 2470
42.6.7 RTC control register (RTC_CR) 2471
42.6.8 RTC privilege mode control register (RTC_PRIVCFGR) 2475
42.6.9 RTC write protection register (RTC_WPR) 2476
42.6.10 RTC calibration register (RTC_CALR) 2477
42.6.11 RTC shift control register (RTC_SHIFTR) 2478
42.6.12 RTC timestamp time register (RTC_TSTR) 2479
42.6.13 RTC timestamp date register (RTC_TSDR) 2480
42.6.14 RTC timestamp subsecond register (RTC_TSSSR) 2481
42.6.15 RTC alarm A register (RTC_ALRMAR) 2481
42.6.16 RTC alarm A subsecond register (RTC_ALRMASSR) 2482
42.6.17 RTC alarm B register (RTC_ALRMBR) 2483
42.6.18 RTC alarm B subsecond register (RTC_ALRMBSSR) 2484
42.6.19 RTC status register (RTC_SR) 2485
42.6.20 RTC masked interrupt status register (RTC_MISR) 2487
42.6.21 RTC status clear register (RTC_SCR) 2488
42.6.22 RTC timestamp on tamper control register (RTC_TAMPTSCR) 2489
42.6.23 RTC timestamp status register (RTC_TSIDR) 2490
42.6.24 RTC alarm A binary mode register (RTC_ALRABINR) 2491
42.6.25 RTC alarm B binary mode register (RTC_ALRBBINR) 2491
42.6.26 RTC register map 2492

43 Tamper and backup registers (TAMP) 2494

43.1 Introduction 2494
43.2 TAMP main features 2494
43.3 TAMP functional description 2495
43.3.1 TAMP block diagram 2495
43.3.2 TAMP pins and internal signals 2496
43.3.3 GPIOs controlled by the RTC and TAMP 2498
43.3.4 TAMP register write protection 2500
43.3.5 Backup registers protection zones 2501
43.3.6 TAMP privilege protection modes 2501
43.3.7 Tamper detection 2502
43.3.8 TAMP backup registers and other device secrets erase 2502
43.3.9 Tamper detection configuration and initialization 2504
43.4 TAMP low-power modes 2511
43.5 TAMP interrupts 2511

43.6 TAMP registers 2511
43.6.1 TAMP control register 1 (TAMP_CR1) 2511
43.6.2 TAMP control register 2 (TAMP_CR2) 2513
43.6.3 TAMP control register 3 (TAMP_CR3) 2516
43.6.4 TAMP filter control register (TAMP_FLTCR) 2517
43.6.5 TAMP active tamper control register 1 (TAMP_ATCR1) 2519
43.6.6 TAMP active tamper seed register (TAMP_ATSEEDR) 2520
43.6.7 TAMP active tamper output register (TAMP_ATOR) 2521
43.6.8 TAMP active tamper control register 2 (TAMP_ATCR2) 2521
43.6.9 TAMP configuration register (TAMP_CFGR) 2523
43.6.10 TAMP privilege configuration register (TAMP_PRIVCFGR) 2524
43.6.11 TAMP interrupt enable register (TAMP_IER) 2525
43.6.12 TAMP status register (TAMP_SR) 2527
43.6.13 TAMP masked interrupt status register (TAMP_MISR) 2529
43.6.14 TAMP status clear register (TAMP_SCR) 2530
43.6.15 TAMP monotonic counter 1 register (TAMP_COUNT1R) 2532
43.6.16 TAMP resources protection configuration register (TAMP_RPCFGR) 2532
43.6.17 TAMP backup x register (TAMP_BKPxR) 2533
43.6.18 TAMP register map 2534

44 Inter-integrated circuit interface (I2C) 2536
44.1 I2C introduction 2536
44.2 I2C main features 2536
44.3 I2C implementation 2537
44.4 I2C functional description 2537
44.4.1 I2C block diagram 2538
44.4.2 I2C pins and internal signals 2538
44.4.3 I2C clock requirements 2539
44.4.4 I2C mode selection 2539
44.4.5 I2C initialization 2540
44.4.6 I2C reset 2544
44.4.7 I2C data transfer 2545
44.4.8 I2C target mode 2547
44.4.9 I2C controller mode 2556
44.4.10 I2C_TIMINGR register configuration examples 2567
44.4.11 SMBus specific features 2569
44.4.12 SMBus initialization 2572

44.4.13 SMBus I2C_TIMEOUTR register configuration examples 2574
44.4.14 SMBus target mode 2574
44.4.15 SMBus controller mode 2578
44.4.16 Wake-up from Stop mode on address match 2581
44.4.17 Error conditions 2582
44.5 I2C in low-power modes 2584
44.6 I2C interrupts 2584
44.7 I2C DMA requests 2585
44.7.1 Transmission using DMA 2585
44.7.2 Reception using DMA 2585
44.8 I2C debug modes 2586
44.9 I2C registers 2586
44.9.1 I2C control register 1 (I2C_CR1) 2586
44.9.2 I2C control register 2 (I2C_CR2) 2589
44.9.3 I2C own address 1 register (I2C_OAR1) 2591
44.9.4 I2C own address 2 register (I2C_OAR2) 2591
44.9.5 I2C timing register (I2C_TIMINGR) 2592
44.9.6 I2C timeout register (I2C_TIMEOUTR) 2593
44.9.7 I2C interrupt and status register (I2C_ISR) 2594
44.9.8 I2C interrupt clear register (I2C_ICR) 2597
44.9.9 I2C PEC register (I2C_PECR) 2598
44.9.10 I2C receive data register (I2C_RXDR) 2598
44.9.11 I2C transmit data register (I2C_TXDR) 2599
44.9.12 I2C register map 2600

45 Improved inter-integrated circuit (I3C) 2601
45.1 I3C introduction 2601
45.2 I3C main features 2601
45.2.1 I3C MIPI® support 2604
45.2.2 I3C without clock stretching by the target 2605
45.3 I3C implementation 2607
45.3.1 I3C instantiation 2607
45.3.2 I3C wake-up from low-power mode(s) 2607
45.3.3 I3C FIFOs 2607
45.3.4 I3C triggers 2607
45.3.5 I3C interrupt(s) 2607

45.4 I3C block diagram 2608
45.5 I3C pins and internal signals 2608
45.6 I3C reset and clocks 2609
45.6.1 I3C reset 2609
45.6.2 I3C clocks and requirements 2609
45.7 I3C peripheral state and programming 2611
45.7.1 I3C peripheral state 2611
45.7.2 I3C controller state and programming sequence 2612
45.7.3 I3C target state and programming sequence 2617
45.8 I3C registers and programming 2624
45.8.1 I3C register set, as controller/target 2624
45.8.2 I3C registers and fields use versus peripheral state, as controller . . 2625
45.8.3 I3C registers and fields usage versus peripheral state, as target . . . 2627
45.9 I3C bus transfers and programming 2629
45.9.1 I3C command set (CCCs), as controller/target 2629
45.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller 2633
45.9.3 I3C broadcast ENTDAA CCC transfer, as controller 2635
45.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller 2635
45.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target 2637
45.9.6 I3C broadcast ENTDAA CCC transfer, as target 2639
45.9.7 I3C broadcast DEFTGTS CCC transfer, as target 2640
45.9.8 I3C broadcast DEFGRPA CCC transfer, as target 2641
45.9.9 I3C direct GETSTATUS CCC response, as target 2642
45.9.10 I3C private read/write transfer, as controller 2643
45.9.11 I3C private read/write transfer, as target 2644
45.9.12 Legacy I2C read/write transfer, as controller 2645
45.9.13 Legacy I2C read/write transfer, as target 2646
45.9.14 I3C IBI transfer, as controller/target 2646
45.9.15 I3C hot-join request transfer, as controller/target 2649
45.9.16 I3C controller-role request transfer, as controller/target 2650
45.10 I3C FIFOs management, as controller 2651
45.10.1 C-FIFO management, as controller 2651
45.10.2 TX-FIFO management, as controller 2652
45.10.3 RX-FIFO management, as controller 2655
45.10.4 S-FIFO management, as controller 2657

45.11 I3C FIFOs management, as target 2659
45.11.1 RX-FIFO management, as target 2659
45.11.2 TX-FIFO management, as target 2662
45.12 I3C error management 2666
45.12.1 Controller error management 2666
45.12.2 Target error management 2668
45.13 I3C wake-up from low-power mode(s) 2669
45.13.1 Wake-up from Stop 2669
45.14 I3C in low-power modes 2672
45.15 I3C interrupts 2673
45.16 I3C registers 2673
45.16.1 I3C message control register (I3C_CR) 2673
45.16.2 I3C message control register [alternate] (I3C_CR) 2675
45.16.3 I3C configuration register (I3C_CFGR) 2677
45.16.4 I3C receive data byte register (I3C_RDR) 2682
45.16.5 I3C receive data word register (I3C_RDWR) 2683
45.16.6 I3C transmit data byte register (I3C_TDR) 2684
45.16.7 I3C transmit data word register (I3C_TDWR) 2685
45.16.8 I3C IBI payload data register (I3C_IBIDR) 2686
45.16.9 I3C target transmit configuration register (I3C_TGTTDR) 2687
45.16.10 I3C status register (I3C_SR) 2688
45.16.11 I3C status error register (I3C_SER) 2689
45.16.12 I3C received message register (I3C_RMR) 2691
45.16.13 I3C event register (I3C_EVR) 2692
45.16.14 I3C interrupt enable register (I3C_IER) 2696
45.16.15 I3C clear event register (I3C_CEVR) 2698
45.16.16 I3C masked interrupt status register (I3C_MISR) 2700
45.16.17 I3C own device characteristics register (I3C_DEVR0) 2701
45.16.18 I3C device x characteristics register (I3C_DEVRx) 2703
45.16.19 I3C maximum read length register (I3C_MAXRLR) 2705
45.16.20 I3C maximum write length register (I3C_MAXWLR) 2706
45.16.21 I3C I2C target configuration register (I3C_I2CCFGR) 2707
45.16.22 I3C timing register 0 (I3C_TIMINGR0) 2708
45.16.23 I3C timing register 1 (I3C_TIMINGR1) 2709
45.16.24 I3C timing register 2 (I3C_TIMINGR2) 2711
45.16.25 I3C bus characteristics register (I3C_BCR) 2713

45.16.26 I3C device characteristics register (I3C_DCR) 2714
45.16.27 I3C get capability register (I3C_GETCAPR) 2715
45.16.28 I3C controller-role capability register (I3C_CRCAPR) 2716
45.16.29 I3C get max data speed register (I3C_GETMXDSR) 2717
45.16.30 I3C extended provisioned ID register (I3C_EPIDR) 2719
45.16.31 I3C register map 2720

46 Universal synchronous/asynchronous receiver transmitter (USART/UART) 2723

46.1 USART introduction 2723
46.2 USART main features 2723
46.3 USART extended features 2724
46.4 USART implementation 2724
46.5 USART functional description 2726
46.5.1 USART block diagram 2726
46.5.2 USART pins and internal signals 2726
46.5.3 USART clocks 2728
46.5.4 USART character description 2728
46.5.5 USART FIFOs and thresholds 2731
46.5.6 USART transmitter 2731
46.5.7 USART receiver 2734
46.5.8 USART baud rate generation 2741
46.5.9 Tolerance of the USART receiver to clock deviation 2743
46.5.10 USART auto baud rate detection 2744
46.5.11 USART multiprocessor communication 2746
46.5.12 USART Modbus communication 2748
46.5.13 USART parity control 2749
46.5.14 USART LIN (local interconnection network) mode 2750
46.5.15 USART synchronous mode 2752
46.5.16 USART single-wire half-duplex communication 2756
46.5.17 USART receiver timeout 2756
46.5.18 USART smartcard mode 2757
46.5.19 USART IrDA SIR ENDEC block 2761
46.5.20 Continuous communication using USART and DMA 2764
46.5.21 RS232 hardware flow control and RS485 driver enable 2766
46.5.22 USART low-power management 2769
46.6 USART in low-power modes 2772

46.7 USART interrupts 2772
46.8 USART registers 2775
46.8.1 USART control register 1 (USART_CR1) 2775
46.8.2 USART control register 1 [alternate] (USART_CR1) 2779
46.8.3 USART control register 2 (USART_CR2) 2782
46.8.4 USART control register 3 (USART_CR3) 2786
46.8.5 USART control register 3 [alternate] (USART_CR3) 2790
46.8.6 USART baud rate register (USART_BRR) 2793
46.8.7 USART guard time and prescaler register (USART_GTPR) 2794
46.8.8 USART receiver timeout register (USART_RTOR) 2795
46.8.9 USART request register (USART_RQR) 2796
46.8.10 USART interrupt and status register (USART_ISR) 2797
46.8.11 USART interrupt and status register [alternate] (USART_ISR) 2803
46.8.12 USART interrupt flag clear register (USART_ICR) 2808
46.8.13 USART receive data register (USART_RDR) 2809
46.8.14 USART transmit data register (USART_TDR) 2810
46.8.15 USART prescaler register (USART_PRESC) 2810
46.8.16 USART register map 2811

47 Low-power universal asynchronous receiver transmitter (LPUART) 2813
47.1 LPUART introduction 2813
47.2 LPUART main features 2813
47.3 LPUART implementation 2814
47.4 LPUART functional description 2816
47.4.1 LPUART block diagram 2816
47.4.2 LPUART pins and internal signals 2817
47.4.3 LPUART clocks 2818
47.4.4 LPUART character description 2818
47.4.5 LPUART FIFOs and thresholds 2820
47.4.6 LPUART transmitter 2820
47.4.7 LPUART receiver 2824
47.4.8 LPUART baud rate generation 2828
47.4.9 Tolerance of the LPUART receiver to clock deviation 2829
47.4.10 LPUART multiprocessor communication 2830
47.4.11 LPUART parity control 2832
47.4.12 LPUART single-wire half-duplex communication 2833

47.4.13 Continuous communication using DMA and LPUART 2833
47.4.14 RS232 hardware flow control and RS485 driver enable 2836
47.4.15 LPUART low-power management 2838
47.5 LPUART in low-power modes 2841
47.6 LPUART interrupts 2842
47.7 LPUART registers 2843
47.7.1 LPUART control register 1 (LPUART_CR1) 2843
47.7.2 LPUART control register 1 [alternate] (LPUART_CR1) 2846
47.7.3 LPUART control register 2 (LPUART_CR2) 2849
47.7.4 LPUART control register 3 (LPUART_CR3) 2851
47.7.5 LPUART control register 3 [alternate] (LPUART_CR3) 2854
47.7.6 LPUART baud rate register (LPUART_BRR) 2856
47.7.7 LPUART request register (LPUART_RQR) 2856
47.7.8 LPUART interrupt and status register (LPUART_ISR) 2857
47.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR) . . . 2862
47.7.10 LPUART interrupt flag clear register (LPUART_ICR) 2865
47.7.11 LPUART receive data register (LPUART_RDR) 2866
47.7.12 LPUART transmit data register (LPUART_TDR) 2866
47.7.13 LPUART prescaler register (LPUART_PRESC) 2867
47.7.14 LPUART register map 2868

48 Serial peripheral interface (SPI) 2870
48.1 SPI introduction 2870
48.2 SPI main features 2870
48.3 SPI implementation 2871
48.4 SPI functional description 2872
48.4.1 SPI block diagram 2872
48.4.2 SPI pins and internal signals 2873
48.4.3 SPI communication general aspects 2874
48.4.4 Communications between one master and one slave 2874
48.4.5 Standard multislave communication 2877
48.4.6 Multimaster communication 2880
48.4.7 Slave select (NSS pin) management 2881
48.4.8 Ready pin (RDY) management 2885
48.4.9 Communication formats 2885
48.4.10 Configuring the SPI 2887

48.4.11 Enabling the SPI . . . 2888
48.4.12 SPI data transmission and reception procedures . . . 2889
48.4.13 Disabling the SPI . . . 2893
48.4.14 Communication using DMA (direct memory addressing) . . . 2894
48.5 SPI specific modes and control . . . 2896
48.5.1 TI mode . . . 2896
48.5.2 SPI error flags . . . 2896
48.5.3 CRC computation . . . 2900
48.6 SPI in low-power modes . . . 2901
48.7 SPI interrupts . . . 2901
48.8 I2S main features . . . 2903
48.9 I2S functional description . . . 2903
48.9.1 I2S general description . . . 2903
48.9.2 Pin sharing with SPI function . . . 2904
48.9.3 Bitfields usable in I2S/PCM mode . . . 2904
48.9.4 Slave and master modes . . . 2905
48.9.5 Supported audio protocols . . . 2905
48.9.6 Additional serial interface flexibility . . . 2911
48.9.7 Startup sequence . . . 2913
48.9.8 Stop sequence . . . 2915
48.9.9 Clock generator . . . 2916
48.9.10 Internal FIFOs . . . 2918
48.9.11 FIFO status flags . . . 2919
48.9.12 Handling of underrun situation . . . 2920
48.9.13 Handling of overrun situation . . . 2921
48.9.14 Frame error detection . . . 2923
48.9.15 DMA interface . . . 2924
48.9.16 Programing examples . . . 2924
48.10 I2S interrupts . . . 2927
48.11 SPI/I2S registers . . . 2928
48.11.1 SPI/I2S control register 1 (SPI_CR1) . . . 2928
48.11.2 SPI/I2S control register 2 (SPI_CR2) . . . 2930
48.11.3 SPI/I2S configuration register 1 (SPI_CFG1) . . . 2930
48.11.4 SPI/I2S configuration register 2 (SPI_CFG2) . . . 2934
48.11.5 SPI/I2S interrupt enable register (SPI_IER) . . . 2936
48.11.6 SPI/I2S status register (SPI_SR) . . . 2937

48.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR) 2940
48.11.8 SPI/I2S transmit data register (SPI_TXDR) 2941
48.11.9 SPI/I2S receive data register (SPI_RXDR) 2941
48.11.10 SPI/I2S polynomial register (SPI_CRCPOLY) 2942
48.11.11 SPI/I2S transmitter CRC register (SPI_TXCRC) 2942
48.11.12 SPI/I2S receiver CRC register (SPI_RXCRC) 2943
48.11.13 SPI/I2S underrun data register (SPI_UDRDR) 2944
48.11.14 SPI/I2S configuration register (SPI_I2SCFGR) 2944
48.11.15 SPI/I2S register map 2946

49 FD controller area network (FDCAN) 2948

49.1 FDCAN introduction 2948
49.2 FDCAN main features 2950
49.3 FDCAN functional description 2951
49.3.1 FDCAN block diagram 2951
49.3.2 FDCAN pins and internal signals 2952
49.3.3 Bit timing 2953
49.3.4 Operating modes 2954
49.3.5 Error management 2963
49.3.6 Message RAM 2964
49.3.7 FIFO acknowledge handling 2973
49.3.8 FDCAN Rx FIFO element 2973
49.3.9 FDCAN Tx buffer element 2975
49.3.10 FDCAN Tx event FIFO element 2977
49.3.11 FDCAN standard message ID filter element 2978
49.3.12 FDCAN extended message ID filter element 2979
49.4 FDCAN registers 2981
49.4.1 FDCAN core release register (FDCAN_CREL) 2981
49.4.2 FDCAN endian register (FDCAN_ENDN) 2981
49.4.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP) 2981
49.4.4 FDCAN test register (FDCAN_TEST) 2982
49.4.5 FDCAN RAM watchdog register (FDCAN_RWD) 2983
49.4.6 FDCAN CC control register (FDCAN_CCCR) 2984
49.4.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP) . 2985
49.4.8 FDCAN timestamp counter configuration register (FDCAN_TSCC) . 2987
49.4.9 FDCAN timestamp counter value register (FDCAN_TSCV) 2987
49.4.10 FDCAN timeout counter configuration register (FDCAN_TOCC) . . . 2988

49.4.11 FDCAN timeout counter value register (FDCAN_TOCV) 2989
49.4.12 FDCAN error counter register (FDCAN_ECR) 2989
49.4.13 FDCAN protocol status register (FDCAN_PSR) 2990
49.4.14 FDCAN transmitter delay compensation register (FDCAN_TDCR) . . 2992
49.4.15 FDCAN interrupt register (FDCAN_IR) 2992
49.4.16 FDCAN interrupt enable register (FDCAN_IE) 2995
49.4.17 FDCAN interrupt line select register (FDCAN_ILS) 2997
49.4.18 FDCAN interrupt line enable register (FDCAN_ILE) 2998
49.4.19 FDCAN global filter configuration register (FDCAN_RXGFC) 2998
49.4.20 FDCAN extended ID and mask register (FDCAN_XIDAM) 3000
49.4.21 FDCAN high-priority message status register (FDCAN_HPMS) 3000
49.4.22 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S) 3001
49.4.23 CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A) 3002
49.4.24 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S) 3002
49.4.25 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A) 3003
49.4.26 FDCAN Tx buffer configuration register (FDCAN_TXBC) 3003
49.4.27 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS) 3004
49.4.28 FDCAN Tx buffer request pending register (FDCAN_TXBRP) 3004
49.4.29 FDCAN Tx buffer add request register (FDCAN_TXBAR) 3005
49.4.30 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR) . . 3006
49.4.31 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO) 3006
49.4.32 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF) . . 3007
49.4.33 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE) 3007
49.4.34 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE) 3008
49.4.35 FDCAN Tx event FIFO status register (FDCAN_TXEFS) 3008
49.4.36 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA) . . . 3009
49.4.37 FDCAN CFG clock divider register (FDCAN_CKDIV) 3009
49.4.38 FDCAN register map 3010

50 USB on-the-go full-speed (OTG_FS) 3014

50.1 OTG_FS introduction 3014
50.2 OTG_FS main features 3015
50.2.1 General features 3015
50.2.2 Host-mode features 3016
50.2.3 Peripheral-mode features 3016

50.3 OTG_FS implementation . . . 3017
50.4 OTG_FS functional description . . . 3018
50.4.1 OTG_FS block diagram . . . 3018
50.4.2 OTG_FS pin and internal signals . . . 3018
50.4.3 OTG_FS core . . . 3019
50.4.4 Embedded full-speed OTG PHY connected to OTG_FS . . . 3019
50.4.5 OTG detections . . . 3020
50.5 OTG_FS dual role device (DRD) . . . 3020
50.5.1 ID line detection . . . 3020
50.6 OTG_FS as a USB peripheral . . . 3021
50.6.1 Peripheral states . . . 3021
50.6.2 Peripheral endpoints . . . 3022
50.7 OTG_FS as a USB host . . . 3024
50.7.1 USB host states . . . 3025
50.7.2 Host channels . . . 3026
50.7.3 Host scheduler . . . 3028
50.8 OTG_FS SOF trigger . . . 3029
50.8.1 Host SOFs . . . 3029
50.8.2 Peripheral SOFs . . . 3029
50.9 OTG_FS low-power modes . . . 3030
50.10 OTG_FS Dynamic update of the OTG_HFIR register . . . 3031
50.11 OTG_FS data FIFOs . . . 3031
50.11.1 Peripheral FIFO architecture . . . 3032
50.11.2 Host FIFO architecture . . . 3033
50.11.3 FIFO RAM allocation . . . 3034
50.12 OTG_FS system performance . . . 3036
50.13 OTG_FS interrupts . . . 3036
50.14 OTG_FS control and status registers . . . 3038
50.14.1 CSR memory map . . . 3038
50.15 OTG_FS registers . . . 3043
50.15.1 OTG control and status register (OTG_GOTGCTL) . . . 3043
50.15.2 OTG interrupt register (OTG_GOTGINT) . . . 3045
50.15.3 OTG AHB configuration register (OTG_GAHBCFG) . . . 3046
50.15.4 OTG USB configuration register (OTG_GUSBCFG) . . . 3047
50.15.5 OTG reset register (OTG_GRSTCTL) . . . 3048
50.15.6 OTG core interrupt register [alternate] (OTG_GINTSTS) . . . 3051

50.15.7 OTG core interrupt register [alternate] (OTG_GINTSTS) 3055
50.15.8 OTG interrupt mask register [alternate] (OTG_GINTMSK) 3059
50.15.9 OTG interrupt mask register [alternate] (OTG_GINTMSK) 3060
50.15.10 OTG receive status debug read [alternate] (OTG_GRXSTSR) 3062
50.15.11 OTG receive status debug read register [alternate] (OTG_GRXSTSR) 3063
50.15.12 OTG status read and pop registers [alternate] (OTG_GRXSTSP) . . 3064
50.15.13 OTG status read and pop registers [alternate] (OTG_GRXSTSP) . . 3065
50.15.14 OTG receive FIFO size register (OTG_GRXFSIZ) 3066
50.15.15 OTG host non-periodic transmit FIFO size register [alternate] (OTG_HNPTXFSIZ) 3067
50.15.16 Endpoint 0 Transmit FIFO size [alternate] (OTG_DIEPTXF0) 3067
50.15.17 OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS) 3068
50.15.18 OTG general core configuration register (OTG_GCCFG) 3069
50.15.19 OTG core ID register (OTG_CID) 3070
50.15.20 OTG core LPM configuration register (OTG_GLPMCFG) 3071
50.15.21 OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ) 3075
50.15.22 OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx) 3075
50.15.23 Host-mode registers 3075
50.15.24 OTG host configuration register (OTG_HCFG) 3076
50.15.25 OTG host frame interval register (OTG_HFIR) 3076
50.15.26 OTG host frame number/frame time remaining register (OTG_HFNUM) 3077
50.15.27 OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS) 3078
50.15.28 OTG host all channels interrupt register (OTG_HAINT) 3079
50.15.29 OTG host all channels interrupt mask register (OTG_HAINTMSK) 3079
50.15.30 OTG host port control and status register (OTG_HPRT) 3080
50.15.31 OTG host channel x characteristics register (OTG_HCCHARx) 3082
50.15.32 OTG host channel x interrupt register (OTG_HCINTx) 3083
50.15.33 OTG host channel x interrupt mask register (OTG_HCINTMSKx) . . 3084
50.15.34 OTG host channel x transfer size register (OTG_HCTSIZx) 3085
50.15.35 Device-mode registers 3086
50.15.36 OTG device configuration register (OTG_DCFG) 3086
50.15.37 OTG device control register (OTG_DCTL) 3087

50.15.38 OTG device status register (OTG_DSTS) 3090
50.15.39 OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK) 3091
50.15.40 OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK) 3092
50.15.41 OTG device all endpoints interrupt register (OTG_DAINT) 3093
50.15.42 OTG all endpoints interrupt mask register (OTG_DAINTMSK) 3094
50.15.43 OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK) 3094
50.15.44 OTG device each endpoint interrupt register (OTG_DEACHINT) . . . 3095
50.15.45 OTG device each endpoint interrupt mask register (OTG_DEACHINTMSK) 3095
50.15.46 OTG device each IN endpoint-1 interrupt mask register (OTG_DIEPEACHM) 3096
50.15.47 OTG device each OUT endpoint-1 interrupt mask register (OTG_DOEPEACHMSK1) 3097
50.15.48 OTG device control IN endpoint 0 control register (OTG_DIEPCTL0) 3098
50.15.49 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx) 3100
50.15.50 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx) 3102
50.15.51 OTG device IN endpoint x interrupt register (OTG_DIEPINTx) 3104
50.15.52 OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0) 3105
50.15.53 OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx) 3106
50.15.54 OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx) . 3106
50.15.55 OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0) 3107
50.15.56 OTG device OUT endpoint x interrupt register (OTG_DOEPINTx) . . 3109
50.15.57 OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0) 3110
50.15.58 OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx) 3111
50.15.59 OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx) 3113
50.15.60 OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx) 3115
50.15.61 OTG power and clock gating control register (OTG_PCGCCTL) . . . 3116
50.15.62 OTG power and clock gating control register 1 (OTG_PCGCCTL1) . 3117

50.15.63 OTG_FS register map . . . 3118
50.16 OTG_FS programming model . . . 3126
50.16.1 Core initialization . . . 3126
50.16.2 Host initialization . . . 3127
50.16.3 Device initialization . . . 3127
50.16.4 Host programming model . . . 3128
50.16.5 Device programming model . . . 3150
50.16.6 Worst case response time . . . 3171
50.16.7 OTG programming model . . . 3173

51 USB Type-C®/USB Power Delivery interface (UCPD) . . . 3174
51.1 UCPD introduction . . . 3174
51.2 UCPD main features . . . 3174
51.3 UCPD implementation . . . 3175
51.4 UCPD functional description . . . 3175
51.4.1 UCPD block diagram . . . 3176
51.4.2 UCPD reset and clocks . . . 3177
51.4.3 Physical layer protocol . . . 3178
51.4.4 UCPD BMC transmitter . . . 3184
51.4.5 UCPD BMC receiver . . . 3186
51.4.6 UCPD Type-C pull-ups (Rp) and pull-downs (Rd) . . . 3187
51.4.7 UCPD Type-C voltage monitoring and de-bouncing . . . 3188
51.4.8 UCPD fast role swap (FRS) . . . 3188
51.4.9 UCPD DMA Interface . . . 3188
51.4.10 Wake-up from Stop mode . . . 3188
51.5 UCPD programming sequences . . . 3189
51.5.1 Initialization phase . . . 3189
51.5.2 Type-C state machine handling . . . 3189
51.5.3 USB PD transmit . . . 3191
51.5.4 USB PD receive . . . 3192
51.6 UCPD low-power modes . . . 3193
51.7 UCPD interrupts . . . 3194
51.8 UCPD registers . . . 3195
51.8.1 UCPD configuration register 1 (UCPD_CFGR1) . . . 3195
51.8.2 UCPD configuration register 2 (UCPD_CFGR2) . . . 3197
51.8.3 UCPD control register (UCPD_CR) . . . 3198

51.8.4 UCPD interrupt mask register (UCPD_IMR) . . . 3200
51.8.5 UCPD status register (UCPD_SR) . . . 3201
51.8.6 UCPD interrupt clear register (UCPD_ICR) . . . 3204
51.8.7 UCPD Tx ordered set type register (UCPD_TX_ORDSETR) . . . 3205
51.8.8 UCPD Tx payload size register (UCPD_TX_PAYSZR) . . . 3206
51.8.9 UCPD Tx data register (UCPD_TXDR) . . . 3206
51.8.10 UCPD Rx ordered set register (UCPD_RX_ORDSETR) . . . 3207
51.8.11 UCPD Rx payload size register (UCPD_RX_PAYSZR) . . . 3207
51.8.12 UCPD receive data register (UCPD_RXDR) . . . 3208
51.8.13 UCPD Rx ordered set extension register 1 (UCPD_RX_ORDEXTR1) . . . 3208
51.8.14 UCPD Rx ordered set extension register 2 (UCPD_RX_ORDEXTR2) . . . 3209
51.8.15 UCPD register map . . . 3209

52 Debug infrastructure . . . 3212

52.1 Introduction . . . 3212
52.2 Debug infrastructure features . . . 3213
52.3 Debug infrastructure functional description . . . 3213
52.3.1 Debug infrastructure block diagram . . . 3213
52.3.2 Debug infrastructure powering, clocking and reset . . . 3214
52.3.3 Security . . . 3215
52.4 Serial-wire and JTAG debug port (SWJ-DP) . . . 3216
52.4.1 Serial wire debug port . . . 3216
52.4.2 JTAG debug port . . . 3219
52.4.3 Debug port registers . . . 3222
52.4.4 Debug port register map . . . 3230
52.5 Access ports . . . 3231
52.5.1 MEM-AP registers . . . 3233
52.5.2 Access port register map . . . 3238
52.6 Trace and debug subsystem functional description . . . 3240
52.6.1 System ROM table . . . 3240
52.6.2 System ROM registers . . . 3243
52.6.3 System ROM register map and reset values . . . 3247
52.6.4 Cross trigger interfaces (CTI) and matrix (CTM) . . . 3248
52.6.5 CTI registers . . . 3251
52.6.6 CTI register map and reset values . . . 3265

52.6.7 Trace funnel (CSTF) . . . 3267
52.6.8 Trace funnel registers . . . 3268
52.6.9 Trace funnel register map and reset values . . . 3276
52.6.10 Embedded trace FIFO (ETF) . . . 3278
52.6.11 ETF registers . . . 3280
52.6.12 ETF register map and reset values . . . 3297
52.6.13 Trace port interface unit (TPIU) . . . 3300
52.6.14 TPIU registers . . . 3301
52.6.15 TPIU register map and reset values . . . 3315
52.7 Serial wire output (SWO) . . . 3317
52.7.1 SWO registers . . . 3318
52.7.2 SWO register map and reset values . . . 3326
52.8 Trace replicator . . . 3328
52.8.1 Trace replicator registers . . . 3329
52.9 Microcontroller debug unit (DBGMCU) . . . 3340
52.9.1 DBGMCU registers . . . 3340
52.9.2 DBGMCU register map and reset values . . . 3355
52.10 Cortex-M7 debug functional description . . . 3358
52.10.1 Cortex-M7 ROM tables . . . 3358
52.10.2 Cortex-M7 CPU ROM registers . . . 3361
52.10.3 Cortex-M7 CPU ROM table register map and reset values . . . 3365
52.10.4 Cortex-M7 PPB ROM registers . . . 3366
52.10.5 Cortex-M7 PPB ROM table register map and reset values . . . 3370
52.10.6 Cortex-M7 data watchpoint and trace unit (DWT) . . . 3371
52.10.7 Cortex-M7 DWT registers . . . 3372
52.10.8 Cortex-M7 DWT register map and reset values . . . 3381
52.10.9 Cortex-M7 instrumentation trace macrocell (ITM) . . . 3383
52.10.10 Cortex-M7 ITM registers . . . 3384
52.10.11 Cortex-M7 ITM register map and reset values . . . 3390
52.10.12 Cortex-M7 breakpoint unit (FPB) . . . 3391
52.10.13 Cortex-M7 FPB registers . . . 3391
52.10.14 Cortex-M7 FPB register map and reset values . . . 3396
52.10.15 Cortex-M7 embedded trace macrocell (ETM) . . . 3397
52.10.16 Cortex-M7 ETM registers . . . 3397
52.10.17 Cortex-M7 ETM register map and reset values . . . 3424
52.10.18 Cortex-M7 cross trigger interface (CTI) . . . 3428

52.11 References for debug infrastructure 3429

53 Device electronic signature 3430

53.1 Unique device ID register (96 bits) 3430

53.2 Package data register 3432

54 Important security notice 3433

55 Revision history 3434

List of tables

Table 1. Availability of security features . . . 107
Table 2. Bus-manager-to-bus-subordinate interconnect . . . 109
Table 3. ASIB configuration . . . 113
Table 4. AMIB configuration . . . 113
Table 5. AXI interconnect register map and reset values . . . 122
Table 6. Memory map and default device memory area attribute . . . 134
Table 7. Peripheral boundary addresses . . . 136
Table 8. SRAM summary . . . 142
Table 9. Boot modes . . . 143
Table 10. DMA channel use (privilege) . . . 148
Table 11. Device cryptographic operations . . . 150
Table 12. Internal tampers in TAMP . . . 151
Table 13. Effect of low-power modes on TAMP . . . 152
Table 14. Main product life cycle transitions . . . 153
Table 15. Macros for RSS services . . . 154
Table 16. RSS lib interface functions . . . 155
Table 17. RAMECC internal input/output signals . . . 164
Table 18. ECC controller mapping . . . 165
Table 19. Effect of low-power modes on RAMECC . . . 166
Table 20. RAMECC interrupt requests . . . 167
Table 21. RAMECC register map and reset values . . . 171
Table 22. FLASH internal input/output signals . . . 177
Table 23. Flash memory organization (2 Mbytes) . . . 180
Table 24. Flash memory organization (1 Mbyte) . . . 181
Table 25. FLASH recommended number of wait states and programming delays . . . 187
Table 26. Flash memory OTP organization . . . 195
Table 27. read-only public data organization . . . 197
Table 28. Flash registers versus swapping option . . . 197
Table 29. Flash operation flags versus swapping option . . . 198
Table 30. Option byte user words organization . . . 204
Table 31. Flash interface register protection summary . . . 209
Table 32. Flash security lifecycle definition . . . 210
Table 33. Option bytes versus lifecycle definition . . . 211
Table 34. Boot level and HDP area protections . . . 212
Table 35. Effect of low-power modes on the embedded flash memory . . . 215
Table 36. Flash interrupt request . . . 221
Table 37. FLASH register map and reset values . . . 251
Table 38. PWR input/output signals connected to package pins or balls . . . 258
Table 39. PWR internal input/output signals . . . 259
Table 40. Operating mode summary . . . 280
Table 41. Sleep mode . . . 288
Table 42. Stop mode operation . . . 289
Table 43. Stop mode . . . 290
Table 44. Standby and Stop flags . . . 292
Table 45. Standby mode . . . 292
Table 46. Low-power modes monitoring pin overview . . . 293
Table 47. GPIO state according to CPU and domain state . . . 293
Table 48. Power control register map and reset values . . . 312

Table 49. RCC input/output signals connected to package pins or balls . . . 316
Table 50. RCC internal input/output signals . . . 316
Table 51. Reset coverage summary . . . 323
Table 52. Reset source identification (RCC_RSR) . . . 324
Table 53. Oscillator states versus system modes . . . 328
Table 54. Oscillator states versus system modes . . . 342
Table 55. VCO frequency and output frequency in integer mode . . . 345
Table 56. VCO frequency and output frequency in fractional mode . . . 346
Table 57. Clock protection summary . . . 351
Table 58. STOPWUCK and STOPKERWUCK description . . . 353
Table 59. HSIKERON and CSIKERON behavior . . . 353
Table 60. Peripheral clock distribution summary . . . 354
Table 61. SDMMC interface clock constraints . . . 368
Table 62. Peripheral clock enabling . . . 380
Table 63. RNG clock enabling . . . 381
Table 64. Interrupt sources and control . . . 384
Table 65. Clock configuration examples with PLLs in integer mode . . . 388
Table 66. Clock configuration examples with PLLs in fractional mode . . . 389
Table 67. RCC register map and reset values . . . 475
Table 68. CRS features . . . 483
Table 69. CRS internal input/output signals . . . 484
Table 70. CRS interconnection . . . 485
Table 71. Effect of low-power modes on CRS . . . 488
Table 72. Interrupt control bits . . . 488
Table 73. CRS register map and reset values . . . 493
Table 74. Port bit configuration . . . 496
Table 75. GPIO register map and reset values . . . 512
Table 76. SBS input/output pins . . . 515
Table 77. SBS internal input/output signals . . . 515
Table 78. SBS boot control . . . 517
Table 79. Analog switches recommended configuration . . . 518
Table 80. SBS register map and reset values . . . 531
Table 81. Peripheral interconnect matrix . . . 533
Table 82. Implementation of HPDMA1 channels . . . 549
Table 83. HPDMA1 in low-power modes . . . 550
Table 84. Programmed HPDMA1 request . . . 550
Table 85. Programmed HPDMA1 request as a block request . . . 554
Table 86. HPDMA1 channel with peripheral early termination . . . 555
Table 87. Programmed HPDMA1 trigger . . . 555
Table 88. Programmed HPDMA source/destination burst . . . 579
Table 89. Programmed data handling . . . 584
Table 90. Effect of low-power modes on HPDMA . . . 607
Table 91. HPDMA interrupt requests . . . 607
Table 92. HPDMA register map and reset values . . . 635
Table 93. GPDMA1 channel implementation . . . 639
Table 94. GPDMA1 wake-up in low-power modes . . . 640
Table 95. Programmed GPDMA1 request . . . 640
Table 96. Programmed GPDMA1 request as a block request . . . 644
Table 97. GPDMA1 channel with peripheral early termination . . . 645
Table 98. Programmed GPDMA1 request with peripheral early termination . . . 645
Table 99. Programmed GPDMA1 trigger . . . 645
Table 100. Programmed GPDMA source/destination burst . . . 669

Table 101. Programmed data handling . . . 674
Table 102. Effect of low-power modes on GPDMA . . . 688
Table 103. GPDMA interrupt requests . . . 688
Table 104. GPDMA register map and reset values . . . 715
Table 105. Neural-ART 11 NPU configuration . . . 719
Table 106. NPU internal signals . . . 721
Table 107. Activation functions example . . . 727
Table 108. Neural-ART 11 functional units memory base addresses . . . 733
Table 109. SysTick clock selection . . . 734
Table 110. NVIC . . . 736
Table 111. EXTI event input configurations and register control . . . 746
Table 112. EXTI event input mapping . . . 749
Table 113. Masking functionality . . . 751
Table 114. Extended interrupt and event controller register map and reset values . . . 764
Table 115. CRC internal input/output signals . . . 767
Table 116. CRC register map and reset values . . . 772
Table 117. CORDIC functions . . . 774
Table 118. Cosine parameters . . . 774
Table 119. Sine parameters . . . 775
Table 120. Phase parameters . . . 775
Table 121. Modulus parameters . . . 776
Table 122. Arctangent parameters . . . 777
Table 123. Hyperbolic cosine parameters . . . 777
Table 124. Hyperbolic sine parameters . . . 778
Table 125. Hyperbolic arctangent parameters . . . 778
Table 126. Natural logarithm parameters . . . 779
Table 127. Natural log scaling factors and corresponding ranges . . . 779
Table 128. Square root parameters . . . 780
Table 129. Square root scaling factors and corresponding ranges . . . 780
Table 130. Precision vs. number of iterations . . . 783
Table 131. CORDIC register map and reset value . . . 790
Table 132. Valid combinations for read and write methods . . . 804
Table 133. FMAC register map and reset values . . . 817
Table 134. HSP features for STM32H7Power . . . 822
Table 135. HSP external pins . . . 823
Table 136. HSP internal signals . . . 824
Table 137. HSP trigger input connections for STM32H7Power . . . 824
Table 138. HSP trigger output connections for STM32H7Power . . . 825
Table 139. HSP break connections on STM32H7Power . . . 826
Table 140. Memory mapping seen from MMC AHB slave . . . 829
Table 141. BSTAT possible values . . . 835
Table 142. BUFFCMB programming examples . . . 842
Table 143. Trigger sources versus priority and task number . . . 854
Table 144. Trigger sources versus priority and task number . . . 855
Table 145. Firmware error number . . . 862
Table 146. HSP interrupt requests . . . 866
Table 147. List of trigger sources . . . 871
Table 148. Snoop signal selection . . . 880
Table 149. Flags and signal sources for break function . . . 881
Table 150. Flags and signal sources for break input and debug functions . . . 882
Table 151. List of locked fields . . . 886
Table 152. Processing list mapping . . . 888

Table 153. List of processing functions . . . 895
Table 154. HSP register map and reset values . . . 952
Table 155. FMC pins . . . 961
Table 156. FMC bank mapping options . . . 964
Table 157. NOR/PSRAM bank selection . . . 964
Table 158. NOR/PSRAM External memory address . . . 964
Table 159. NAND memory mapping and timing registers . . . 965
Table 160. NAND bank selection . . . 965
Table 161. SDRAM bank selection . . . 965
Table 162. SDRAM address mapping . . . 966
Table 163. SDRAM address mapping with 8-bit data bus width . . . 966
Table 164. SDRAM address mapping with 16-bit data bus width . . . 967
Table 165. SDRAM address mapping with 32-bit data bus width . . . 968
Table 166. Programmable NOR/PSRAM access parameters . . . 969
Table 167. Non-multiplexed I/O NOR flash memory . . . 970
Table 168. 16-bit multiplexed I/O NOR flash memory . . . 970
Table 169. Non-multiplexed I/Os PSRAM/SRAM . . . 971
Table 170. 16-Bit multiplexed I/O PSRAM . . . 971
Table 171. NOR flash/PSRAM: Example of supported memories and transactions . . . 972
Table 172. FMC_BCRx bitfields (mode 1) . . . 975
Table 173. FMC_BTRx bitfields (mode 1) . . . 976
Table 174. FMC_BCRx bitfields (mode A) . . . 978
Table 175. FMC_BTRx bitfields (mode A) . . . 979
Table 176. FMC_BWTRx bitfields (mode A) . . . 979
Table 177. FMC_BCRx bitfields (mode 2/B) . . . 981
Table 178. FMC_BTRx bitfields (mode 2/B) . . . 982
Table 179. FMC_BWTRx bitfields (mode 2/B) . . . 982
Table 180. FMC_BCRx bitfields (mode C) . . . 984
Table 181. FMC_BTRx bitfields (mode C) . . . 985
Table 182. FMC_BWTRx bitfields (mode C) . . . 985
Table 183. FMC_BCRx bitfields (mode D) . . . 987
Table 184. FMC_BTRx bitfields (mode D) . . . 987
Table 185. FMC_BWTRx bitfields (mode D) . . . 988
Table 186. FMC_BCRx bitfields (Muxed mode) . . . 990
Table 187. FMC_BTRx bitfields (Muxed mode) . . . 990
Table 188. FMC_BCRx bitfields (Synchronous multiplexed read mode) . . . 995
Table 189. FMC_BTRx bitfields (Synchronous multiplexed read mode) . . . 996
Table 190. FMC_BCRx bitfields (Synchronous multiplexed write mode) . . . 997
Table 191. FMC_BTRx bitfields (Synchronous multiplexed write mode) . . . 998
Table 192. Programmable NAND flash access parameters . . . 1008
Table 193. 8-bit NAND flash memory . . . 1008
Table 194. 16-bit NAND flash memory . . . 1009
Table 195. Supported memories and transactions . . . 1009
Table 196. ECC result relevant bits . . . 1019
Table 197. SDRAM signals . . . 1020
Table 198. FMC register map . . . 1036
Table 199. XSPI implementation . . . 1040
Table 200. XSPI input/output pins . . . 1042
Table 201. XSPI internal signals . . . 1042
Table 202. Command/address phase description . . . 1051
Table 203. OctaRAM command address bit assignment

(based on 64 Mb OctaRAM) . . . 1062
Table 204. Address alignment cases . . . 1069
Table 205. XSPI interrupt requests . . . 1071
Table 206. XSPI register map and reset values . . . 1094
Table 207. SDMMC features . . . 1097
Table 208. SDMMC operation modes SD and SDIO . . . 1100
Table 209. SDMMC operation modes e•MMC . . . 1100
Table 210. SDMMC internal input/output signals . . . 1101
Table 211. SDMMC pins . . . 1102
Table 212. SDMMC Command and data phase selection . . . 1103
Table 213. Command token format . . . 1109
Table 214. Short response with CRC token format . . . 1110
Table 215. Short response without CRC token format . . . 1110
Table 216. Long response with CRC token format . . . 1110
Table 217. Specific Commands overview . . . 1111
Table 218. Command path status flags . . . 1112
Table 219. Command path error handling . . . 1112
Table 220. Data token format . . . 1120
Table 221. Data path status flags and clear bits . . . 1120
Table 222. Data path error handling . . . 1122
Table 223. Data FIFO access . . . 1123
Table 224. Transmit FIFO status flags . . . 1124
Table 225. Receive FIFO status flags . . . 1125
Table 226. AHB and SDMMC_CK clock frequency relation . . . 1130
Table 227. SDIO special operation control . . . 1130
Table 228. 4-bit mode Start, interrupt, and CRC-status Signaling detection . . . 1134
Table 229. CMD12 use cases . . . 1139
Table 230. SDMMC interrupts . . . 1153
Table 231. Response type and SDMMC_RESPxR registers . . . 1160
Table 232. SDMMC register map . . . 1176
Table 233. STM32H7Pxxx features . . . 1179
Table 234. DLYB internal input/output signals . . . 1180
Table 235. DLYB interconnection . . . 1180
Table 236. Delay block control . . . 1181
Table 237. DLYB register map and reset values . . . 1184
Table 238. ADC features . . . 1187
Table 239. ADC input/output pins . . . 1189
Table 240. ADC internal input/output signals . . . 1189
Table 241. ADC1/2/3/4 interconnection . . . 1189
Table 242. Configuring the trigger polarity for regular external triggers . . . 1215
Table 243. Configuring the trigger polarity for injected external triggers . . . 1215
Table 244. TSAR timings depending on resolution . . . 1225
Table 245. Offset computation versus data resolution . . . 1230
Table 246. 12-bit data formats . . . 1232
Table 247. Numerical examples for 32-bit or 16-bit format (POSSPFF = 0) . . . 1233
Table 248. Analog window watchdog features . . . 1241
Table 249. Analog watchdog 1 channel selection . . . 1242
Table 250. Analog watchdog 1, 2, 3 comparison . . . 1243
Table 251. Maximum output results versus N and M . . . 1248
Table 252. Oversampler operating mode summary . . . 1252
Table 253. DELAY bits versus ADC resolution . . . 1260
Table 254. ADC interrupts . . . 1276

Table 255. ADC register map and reset values for each ADC . . . 1317
Table 256. ADC register map and reset values (master and slave ADC common registers) . . . 1321
Table 257. DTS internal input/output signals . . . 1323
Table 258. Sampling time configuration . . . 1326
Table 259. Trigger configuration . . . 1327
Table 260. Temperature sensor behavior in low-power modes . . . 1329
Table 261. DTS register map and reset values . . . 1337
Table 262. DAC features . . . 1340
Table 263. DAC input/output pins . . . 1342
Table 264. DAC internal input/output signals . . . 1342
Table 265. DAC interconnection . . . 1343
Table 266. Data format (case of 12-bit data) . . . 1346
Table 267. HFSEL description . . . 1347
Table 268. Sample and refresh timings . . . 1357
Table 269. Channel output modes summary . . . 1358
Table 270. Effect of low-power modes on DAC . . . 1366
Table 271. DAC interrupts . . . 1367
Table 272. DAC register map and reset values . . . 1387
Table 273. VREFBUF typical values . . . 1390
Table 274. VREF buffer modes . . . 1391
Table 275. VREFBUF trimming data . . . 1392
Table 276. VREFBUF register map and reset values . . . 1395
Table 277. COMP implementation . . . 1397
Table 278. Window paired comparators . . . 1397
Table 279. COMP input/output pins . . . 1398
Table 280. COMP input/output internal signals . . . 1399
Table 281. COMP_INP non-inverting input signals (COMP1, COMP2, COMP3, COMP4) . . . 1399
Table 282. COMP_INP non-inverting input signals (COMP5, COMP6, COMP7, COMP8) . . . 1399
Table 283. COMP_INM inverting input signals (COMP1, COMP2, COMP3, COMP4) . . . 1399
Table 284. COMP_INM inverting input signals (COMP5, COMP6, COMP7, COMP8) . . . 1400
Table 285. COMP blanking internal input signals (COMP1, COMP2, COMP3, COMP4) . . . 1400
Table 286. COMP blanking internal input signals (COMP5, COMP6, COMP7, COMP8) . . . 1401
Table 287. Comparator behavior in the low-power modes . . . 1405
Table 288. Interrupt control bits . . . 1406
Table 289. Interrupt control bits . . . 1407
Table 290. COMP register map and reset values . . . 1413
Table 291. ADF features . . . 1415
Table 292. ADF external pins . . . 1416
Table 293. ADF internal signals . . . 1416
Table 294. ADF trigger connections . . . 1417
Table 295. MDF ADC data connections . . . 1417
Table 296. Control of the common clock generation . . . 1423
Table 297. Clock constraints with respect to the incoming stream . . . 1424
Table 298. Data size according to CIC order and CIC decimation values . . . 1429
Table 299. Possible gain values . . . 1430
Table 300. Recommended maximum gain values versus CIC decimation ratios . . . 1432
Table 301. Most common microphone settings . . . 1433
Table 302. HPF 3 dB cut-off frequency examples . . . 1435
Table 303. ANSLP values versus FRSIZE and sampling rates . . . 1448
Table 304. Threshold values according SNTHR . . . 1449

Table 305. Register protection summary . . . 1455
Table 306. Effect of low-power modes on ADF . . . 1456
Table 307. ADF interrupt requests . . . 1457
Table 308. Examples of ADF settings for microphone capture . . . 1458
Table 309. Programming sequence (CIC4) . . . 1459
Table 310. Programming sequence (CIC5) . . . 1460
Table 311. Output signal levels . . . 1465
Table 312. ADF register map and reset values . . . 1485
Table 313. RNG internal input/output signals . . . 1489
Table 314. RNG interrupt requests . . . 1498
Table 315. RNG configurations . . . 1499
Table 316. Additional health test configurations . . . 1499
Table 317. Configuration selection . . . 1499
Table 318. RNG register map and reset map . . . 1507
Table 319. HASH internal input/output signals . . . 1509
Table 320. Information on supported hash algorithms . . . 1510
Table 321. Hash processor outputs . . . 1513
Table 322. Processing time (in clock cycle) . . . 1519
Table 323. HASH interrupt requests . . . 1520
Table 324. HASH1 register map and reset values . . . 1528
Table 325. HRTIM inputs/outputs summary . . . 1534
Table 326. HRTIM1/HRTIM2 external events mapping and associated features . . . 1536
Table 327. HRTIM1/HRTIM2 update enable inputs and sources . . . 1538
Table 328. HRTIM1/HRTIM2 Burst mode clock sources . . . 1538
Table 329. HRTIM1/HRTIM2 Burst mode trigger source . . . 1538
Table 330. HRTIM1/HRTIM2 internal synchronization sources . . . 1538
Table 331. HRTIM1/HRTIM2 remote update disable interconnect . . . 1538
Table 332. HRTIM1/HRTIM2 system fault sources selection . . . 1539
Table 333. HRTIM1/HRTIM2 fault inputs . . . 1539
Table 334. HRTIM1/HRTIM2 external trigger inputs . . . 1540
Table 335. HRTIM1/HRTIM2 DAC triggers connections . . . 1540
Table 336. Timer resolution and min. PWM frequency for f_{HRTIM} = 520 MHz . . . 1541
Table 337. Period and compare registers min and max values . . . 1543
Table 338. Timer operating modes . . . 1544
Table 339. Events mapping across timer A to F . . . 1549
Table 340. Interleaved mode selection . . . 1550
Table 341. Compare 1..3 values in interleaved mode . . . 1551
Table 342. Deadtime resolution and max absolute values . . . 1563
Table 343. Roll-over event destination and mode programming . . . 1576
Table 344. EExFLTR[4:0] codes depending on UDM bit setting . . . 1578
Table 345. External events features . . . 1582
Table 346. External events selection . . . 1582
Table 347. Output set/reset latency and jitter versus external event operating mode . . . 1584
Table 348. Filtering signals mapping per timer . . . 1587
Table 349. Windowing signals mapping per timer (EEFLTR[3:0] = 1111) . . . 1589
Table 350. HRTIM preloadable control registers and associated update sources . . . 1599
Table 351. Master timer update event propagation . . . 1604
Table 352. TIMx update event propagation . . . 1604
Table 353. Reset events able to generate an update . . . 1606
Table 354. Update event propagation for a timer reset . . . 1606
Table 355. Output state programming, x= A..F, y = 1 or 2 . . . 1608
Table 356. Timer output programming for burst mode . . . 1611

Table 357. Sampling rate and filter length vs FLTFxF[3:0] and clock setting . . . 1621
Table 358. Fault input blanking events . . . 1622
Table 359. Fault 1..6 counter reset source . . . 1623
Table 360. Effect of sync event versus timer operating modes . . . 1628
Table 361. DAC dual channel trigger example . . . 1636
Table 362. HRTIM interrupt summary . . . 1639
Table 363. HRTIM DMA request summary . . . 1641
Table 364. HRTIM Register map and reset values – master timer . . . 1816
Table 365. HRTIM register map and reset values – TIMx (x= A..F) . . . 1817
Table 366. HRTIM register map and reset values – common functions . . . 1822
Table 367. TIM input/output pins . . . 1829
Table 368. TIM internal input/output signals . . . 1829
Table 369. Interconnect to the tim_ti1 input multiplexer . . . 1831
Table 370. Interconnect to the tim_ti2 input multiplexer . . . 1831
Table 371. Interconnect to the tim_ti3 input multiplexer . . . 1831
Table 372. Interconnect to the tim_ti4 input multiplexer . . . 1831
Table 373. Internal trigger connection . . . 1831
Table 374. Interconnect to the tim_etr input multiplexer . . . 1832
Table 375. Interconnect to the tim_brk_cmp multiplexer . . . 1833
Table 376. Timer break2 interconnect . . . 1834
Table 377. Interconnect to tim_ocref_clr multiplexer . . . 1834
Table 378. System break interconnect . . . 1835
Table 379. CCR and ARR register change dithering pattern . . . 1868
Table 380. CCR register change dithering pattern in center-aligned PWM mode . . . 1869
Table 381. Asymmetric PWM modes . . . 1871
Table 382. Combined PWM mode overview . . . 1874
Table 383. Behavior of timer outputs versus tim_brk/tim_brk2 inputs . . . 1885
Table 384. Break protection disarming conditions . . . 1889
Table 385. Counting direction versus encoder signals (CC1P = CC2P = 0) . . . 1901
Table 386. Counting direction versus encoder signals and polarity settings . . . 1905
Table 387. DMA request . . . 1930
Table 388. Effect of low-power modes on TIM1/TIM8/TIM20 . . . 1931
Table 389. Interrupt requests . . . 1932
Table 390. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature . . . 1964
Table 391. TIM1/TIM8/TIM20 register map and reset values . . . 2002
Table 392. STM32H7Pxxx general purpose timers . . . 2007
Table 393. TIM input/output pins . . . 2009
Table 394. TIM internal input/output signals . . . 2009
Table 395. Interconnect to the tim_ti1 input multiplexer . . . 2010
Table 396. Interconnect to the tim_ti2 input multiplexer . . . 2010
Table 397. Interconnect to the tim_ti3 input multiplexer . . . 2011
Table 398. Interconnect to the tim_ti4 input multiplexer . . . 2011
Table 399. TIMx internal trigger connection . . . 2012
Table 400. Interconnect to the tim_etr input multiplexer . . . 2012
Table 401. Interconnect to the tim_ocref_clr input multiplexer . . . 2013
Table 402. CCR and ARR register change dithering pattern . . . 2046
Table 403. CCR register change dithering pattern in center-aligned PWM mode . . . 2047
Table 404. Asymmetric PWM modes . . . 2048
Table 405. Combined PWM mode overview . . . 2049
Table 406. Counting direction versus encoder signals(CC1P = CC2P = 0) . . . 2060
Table 407. Counting direction versus encoder signals and polarity settings . . . 2065

Table 408. DMA request . . . 2094
Table 409. Effect of low-power modes on TIM2/TIM3/TIM4/TIM5 . . . 2094
Table 410. Interrupt requests . . . 2095
Table 411. Output control bit for standard tim_ocx channels . . . 2119
Table 412. TIM2/TIM3/TIM4/TIM5 register map and reset values . . . 2131
Table 413. TIM internal input/output signals . . . 2135
Table 414. TIM_ARR register change dithering pattern . . . 2145
Table 415. DMA request . . . 2147
Table 416. Effect of low-power modes on TIM6/TIM7 . . . 2147
Table 417. Interrupt request . . . 2147
Table 418. TIM6/TIM7 register map and reset values . . . 2155
Table 419. TIM input/output pins . . . 2159
Table 420. TIM internal input/output signals . . . 2159
Table 421. Interconnect to the tim_ti1 input multiplexer . . . 2160
Table 422. Interconnect to the tim_ti2 input multiplexer . . . 2160
Table 423. TIMx internal trigger connection . . . 2161
Table 424. CCR and ARR register change dithering pattern . . . 2179
Table 425. Effect of low-power modes on TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 . . . 2191
Table 426. Interrupt requests . . . 2191
Table 427. Output control bit for standard tim_ocx channels . . . 2205
Table 428. TIM9/TIM12 register map and reset values . . . 2209
Table 429. Output control bit for standard tim_ocx channels . . . 2218
Table 430. TIM10/TIM11/TIM13/TIM14 register map and reset values . . . 2221
Table 431. TIM input/output pins . . . 2226
Table 432. TIM internal input/output signals . . . 2227
Table 433. Interconnect to the tim_ti1 input multiplexer . . . 2228
Table 434. Interconnect to the tim_ti2 input multiplexer . . . 2228
Table 435. TIMx internal trigger connection . . . 2228
Table 436. Timer break interconnect . . . 2229
Table 437. System break interconnect . . . 2230
Table 438. Interconnect to the ocref_clr input multiplexer . . . 2230
Table 439. CCR and ARR register change dithering pattern . . . 2261
Table 440. CCR register change dithering pattern in center-aligned PWM mode . . . 2262
Table 441. Asymmetric PWM modes . . . 2263
Table 442. Break protection disarming conditions . . . 2275
Table 443. Counting direction versus encoder signals (CC1P = CC2P = 0) . . . 2283
Table 444. Counting direction versus encoder signals and polarity settings . . . 2287
Table 445. DMA request . . . 2300
Table 446. Effect of low-power modes on TIM15/TIM16/TIM17 . . . 2300
Table 447. Interrupt requests . . . 2301
Table 448. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15) . . . 2321
Table 449. TIM15 register map and reset values . . . 2339
Table 450. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) . . . 2353
Table 451. TIM16/TIM17 register map and reset values . . . 2370
Table 452. STM32H7Pxxx LPTIM features . . . 2374
Table 453. LPTIM1/2/3 input/output pins . . . 2376
Table 454. LPTIM4/5 input/output pins . . . 2376
Table 455. LPTIM1/2/3 internal signals . . . 2377
Table 456. LPTIM4/5 internal signals . . . 2377
Table 457. LPTIM1/2/3/4/5 external trigger connections . . . 2378

Table 458. LPTIM1/2/3 input 1 connections . . . 2378
Table 459. LPTIM1/2/3 input 2 connections . . . 2378
Table 460. LPTIM1/2/3 input capture 1 connections . . . 2378
Table 461. LPTIM1 input capture 2 connections . . . 2379
Table 462. LPTIM2 input capture 2 connections . . . 2379
Table 463. LPTIM3 input capture 2 connections . . . 2379
Table 464. Prescaler division ratios . . . 2381
Table 465. Encoder counting scenarios . . . 2388
Table 466. Input capture Glitch filter latency (in counter step unit) . . . 2392
Table 467. Interrupt events . . . 2397
Table 468. LPTIM register map and reset values . . . 2421
Table 469. IWDG features . . . 2424
Table 470. IWDG delays versus actions . . . 2425
Table 471. IWDG internal input/output signals . . . 2426
Table 472. Effect of low power modes on IWDG . . . 2431
Table 473. IWDG interrupt request . . . 2433
Table 474. IWDG register map and reset values . . . 2439
Table 475. WWDG features . . . 2440
Table 476. WWDG internal input/output signals . . . 2441
Table 477. WWDG register map and reset values . . . 2446
Table 478. RTC input/output pins . . . 2449
Table 479. RTC internal input/output signals . . . 2449
Table 480. RTC interconnection . . . 2450
Table 481. RTC_OUT mapping . . . 2450
Table 482. Effect of low-power modes on RTC . . . 2464
Table 483. RTC pins functionality over modes . . . 2464
Table 484. Interrupt requests . . . 2465
Table 485. RTC register map and reset values . . . 2492
Table 486. TAMP input/output pins . . . 2496
Table 487. TAMP internal input/output signals . . . 2496
Table 488. TAMP interconnection . . . 2497
Table 489. RTC pin PC13 configuration . . . 2499
Table 490. Backup registers tamper protection . . . 2503
Table 491. Other device resource x tamper protection . . . 2504
Table 492. Active tamper output change period . . . 2507
Table 493. Minimum ATPER value . . . 2508
Table 494. Active tamper filtered pulse duration . . . 2509
Table 495. Effect of low-power modes on TAMP . . . 2511
Table 496. TAMP pins functionality over modes . . . 2511
Table 497. Interrupt requests . . . 2511
Table 498. TAMP register map and reset values . . . 2534
Table 499. I2C implementation . . . 2537
Table 500. I2C input/output pins . . . 2538
Table 501. I2C internal input/output signals . . . 2539
Table 502. Comparison of analog and digital filters . . . 2541
Table 503. I²C-bus and SMBus specification data setup and hold times . . . 2543
Table 504. I2C configuration . . . 2547
Table 505. I²C-bus and SMBus specification clock timings . . . 2558
Table 506. Timing settings for f_{I2CCLK} of 8 MHz . . . 2568
Table 507. Timing settings for f_{I2CCLK} of 16 MHz . . . 2568
Table 508. Timing settings for f_{I2CCLK} of 48 MHz . . . 2569
Table 509. SMBus timeout specifications . . . 2571

Table 510. SMBus with PEC configuration . . . 2573
Table 511. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms . . . 2574
Table 512. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms . . . 2574
Table 513. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs . . . 2574
Table 514. Effect of low-power modes to I2C . . . 2584
Table 515. I2C interrupt requests . . . 2584
Table 516. I2C register map and reset values . . . 2600
Table 517. I3C peripheral controller/target features versus MIPI v1.1 . . . 2604
Table 518. I3C instantiation . . . 2607
Table 519. I3C wake-up . . . 2607
Table 520. I3C FIFOs implementation . . . 2607
Table 521. I3C interrupt(s) . . . 2607
Table 522. I3C input/output pins . . . 2608
Table 523. I3C internal input/output signals . . . 2608
Table 524. I3C register usage . . . 2624
Table 525. I3C registers/fields usage versus controller state . . . 2625
Table 526. I3C registers/fields usage versus target state . . . 2627
Table 527. List of supported I3C CCCs, as controller/target . . . 2630
Table 528. I3C controller error management . . . 2666
Table 529. I3C target error management . . . 2668
Table 530. Effect of low-power modes . . . 2672
Table 531. I3C register map and reset values . . . 2720
Table 532. Instance implementation on STM32H7Pxxx . . . 2724
Table 533. USART/LPUART features . . . 2725
Table 534. USART/UART input/output pins . . . 2727
Table 535. USART internal input/output signals . . . 2728
Table 536. Noise detection from sampled data . . . 2740
Table 537. Tolerance of the USART receiver when BRR [3:0] = 0000 . . . 2744
Table 538. Tolerance of the USART receiver when BRR[3:0] is different from 0000 . . . 2744
Table 539. USART frame formats . . . 2749
Table 540. Effect of low-power modes on the USART . . . 2772
Table 541. USART interrupt requests . . . 2773
Table 542. USART register map and reset values . . . 2811
Table 543. Instance implementation on STM32H7Pxxx . . . 2814
Table 544. USART/LPUART features . . . 2814
Table 545. LPUART input/output pins . . . 2817
Table 546. LPUART internal input/output signals . . . 2817
Table 547. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz . . . 2828
Table 548. Tolerance of the LPUART receiver . . . 2829
Table 550. Effect of low-power modes on the LPUART . . . 2841
Table 551. LPUART interrupt requests . . . 2842
Table 552. LPUART register map and reset values . . . 2868
Table 553. SPI features . . . 2871
Table 554. SPI/I2S input/output pins . . . 2874
Table 555. SPI internal input/output signals . . . 2874
Table 556. Effect of low-power modes on the SPI . . . 2901
Table 557. SPI wake-up and interrupt requests . . . 2902
Table 558. Bitfields usable in PCM/I2S mode . . . 2904
Table 559. WS and CK level before SPI/I2S is enabled when AFCNTR = 1 . . . 2913
Table 560. Serial data line swapping . . . 2913
Table 561. CLKGEN programming examples for usual I2S frequencies . . . 2918
Table 562. I2S interrupt requests . . . 2927

Table 563. SPI register map and reset values . . . 2946
Table 564. CAN subsystem I/O signals . . . 2952
Table 565. CAN subsystem I/O pins . . . 2953
Table 566. DLC coding in FDCAN . . . 2956
Table 567. Possible configurations for frame transmission . . . 2970
Table 568. Rx FIFO element . . . 2973
Table 569. Rx FIFO element description . . . 2973
Table 570. Tx buffer and FIFO element . . . 2975
Table 571. Tx buffer element description . . . 2975
Table 572. Tx event FIFO element . . . 2977
Table 573. Tx event FIFO element description . . . 2977
Table 574. Standard message ID filter element . . . 2978
Table 575. Standard message ID filter element field description . . . 2979
Table 576. Extended message ID filter element . . . 2979
Table 577. Extended message ID filter element field description . . . 2980
Table 578. FDCAN register map and reset values . . . 3010
Table 579. OTG_FS speeds supported . . . 3014
Table 580. OTG_FS implementation . . . 3017
Table 581. OTG_FS input/output pins . . . 3018
Table 582. OTG_FS input/output signals . . . 3019
Table 583. Compatibility of STM32 low power modes with the OTG . . . 3030
Table 584. Core global control and status registers (CSRs) . . . 3038
Table 585. Host-mode control and status registers (CSRs) . . . 3039
Table 586. Device-mode control and status registers . . . 3040
Table 587. Data FIFO (DFIFO) access register map . . . 3042
Table 588. Power and clock gating control and status registers . . . 3042
Table 589. TRDT values . . . 3048
Table 590. Minimum duration for soft disconnect . . . 3089
Table 591. OTG_FS register map and reset values . . . 3118
Table 592. UCPD implementation . . . 3175
Table 593. UCPD signals on pins . . . 3176
Table 594. UCPD internal signals . . . 3177
Table 595. 4b5b symbol encoding table . . . 3178
Table 596. Ordered sets . . . 3180
Table 597. Validation of ordered sets . . . 3180
Table 598. Data size . . . 3181
Table 599. Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx . . . 3189
Table 600. Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2) . . . 3191
Table 601. Effect of low power modes on the UCPD . . . 3193
Table 602. UCPD interrupt requests . . . 3194
Table 603. UCPD register map and reset values . . . 3209
Table 604. Authentication signal states . . . 3216
Table 605. Packet request . . . 3217
Table 606. ACK response . . . 3217
Table 607. Data transfer . . . 3217
Table 608. JTAG-DP data registers . . . 3220
Table 609. Debug port registers . . . 3222
Table 610. Debug port register map and reset values . . . 3230
Table 611. MEM-AP registers . . . 3233
Table 612. Access port register map and reset values . . . 3238
Table 613. System ROM table . . . 3241
Table 614. System ROM table register map and reset values . . . 3247

Table 615. System CTI inputs . . . 3249
Table 616. System CTI outputs . . . 3249
Table 617. System CTI-2 inputs . . . 3249
Table 618. Cortex-M7 CTI inputs . . . 3250
Table 619. Cortex-M7 CTI outputs . . . 3250
Table 620. CTI register map and reset values . . . 3265
Table 621. CSTF register map and reset values . . . 3276
Table 622. ETF register map and reset values . . . 3297
Table 623. TPIU register map and reset values . . . 3315
Table 624. SWO register map and reset values . . . 3326
Table 625. Replicator register map and reset values . . . 3338
Table 626. DBGMCU register map and reset values . . . 3355
Table 627. Cortex-M7 CPU ROM table . . . 3358
Table 628. Cortex-M7 PPB ROM table . . . 3359
Table 629. Cortex-M7 CPU ROM table register map and reset values . . . 3365
Table 630. Cortex-M7 PPB ROM table register map and reset values . . . 3370
Table 631. Cortex-M7 DWT register map and reset values . . . 3381
Table 632. Cortex-M7 ITM register map and reset values . . . 3390
Table 633. Cortex-M7 FPB register map and reset values . . . 3396
Table 634. Cortex-M7 ETM register map and reset values . . . 3424
Table 635. Document revision history . . . 3434

List of figures

Figure 1. System architecture . . . 108
Figure 2. AXI interconnect . . . 112
Figure 3. Memory map . . . 133
Figure 4. User flash memory HDP area principle . . . 147
Figure 5. Device life cycle security . . . 153
Figure 6. RAM ECC controller implementation schematic . . . 163
Figure 7. Connection between RAM ECC controller and RAMECC monitoring unit . . . 164
Figure 8. FLASH block diagram . . . 176
Figure 9. Detailed FLASH architecture . . . 178
Figure 10. Embedded flash memory organization . . . 179
Figure 11. Embedded flash memory usage . . . 182
Figure 12. Flash decommissioning options . . . 183
Figure 13. FLASH read pipeline architecture . . . 186
Figure 14. FLASH write pipeline architecture . . . 189
Figure 15. FLASH stateful initialization . . . 199
Figure 16. Life cycle management scheme . . . 210
Figure 17. Power control block diagram . . . 258
Figure 18. Power supply overview . . . 261
Figure 19. Supply configurations . . . 262
Figure 20. Device startup with VCORE supplied from integrated SMPS or voltage regulator . . . 264
Figure 21. Device startup sequence with VCORE supplied in Bypass mode from external regulator . . . 265
Figure 22. Backup domain . . . 270
Figure 23. Power-on reset/power-down reset waveform . . . 272
Figure 24. BOR thresholds . . . 273
Figure 25. PVD thresholds . . . 274
Figure 26. AVD thresholds . . . 275
Figure 27. VBAT thresholds . . . 276
Figure 28. Temperature thresholds . . . 277
Figure 29. VCORE overvoltage protection . . . 278
Figure 30. V_{CORE} voltage scaling versus system power modes . . . 281
Figure 31. Dynamic voltage scaling in Run mode . . . 283
Figure 32. Dynamic voltage scaling behavior in Stop mode . . . 284
Figure 33. Dynamic voltage scaling from Standby mode . . . 285
Figure 34. RCC block diagram . . . 315
Figure 35. Simplified reset circuit . . . 319
Figure 36. Boot sequences versus system states . . . 325
Figure 37. Top-level clock tree . . . 327
Figure 38. HSE/LSE clock source . . . 329
Figure 39. HSE clock generation . . . 329
Figure 40. LSE clock generation . . . 331
Figure 41. HSI calibration flow . . . 335
Figure 42. CSI calibration flow . . . 336
Figure 43. HSI48 calibration flow . . . 337
Figure 44. LSE and HSE CSS function . . . 338
Figure 45. PLL block diagram . . . 344
Figure 46. Spread-spectrum modulation . . . 347

Figure 47. Triangular waveform generator . . . 348
Figure 48. Core and bus clock generation . . . 350
Figure 49. Key signals controlling low-power modes . . . 352
Figure 50. Clock distribution for HSP and AHBSRAM3 . . . 360
Figure 51. Clock distribution for ADF . . . 361
Figure 52. Clock distribution for SPIs and SPI/I2S . . . 362
Figure 53. Clock distribution for I2C[3:1] and I3C1 . . . 363
Figure 54. Clock distribution for UARTs, LPUART1 and USARTs . . . 365
Figure 55. Clock distribution for FDCAN . . . 366
Figure 56. Clock distribution for FMC and XSPI . . . 367
Figure 57. Clock distribution for SDMMC and DB_SDMMC . . . 368
Figure 58. Clock distribution for USB and UCPD . . . 369
Figure 59. Clock distribution for ADCs and DACs . . . 370
Figure 60. Clock distribution for RNG . . . 371
Figure 61. Clock distribution for TIM and HRTIM . . . 372
Figure 62. Clock distribution for LPTIMs . . . 373
Figure 63. Clock distribution for RTC . . . 374
Figure 64. Clock distribution for WWDG and IWDG . . . 375
Figure 65. Clock distribution for DBG and trace . . . 375
Figure 66. Kernel clock switching . . . 378
Figure 67. Peripheral kernel clock enable logic details . . . 379
Figure 68. CRS block diagram . . . 484
Figure 69. CRS counter behavior . . . 486
Figure 70. Structure of three-volt or five-volt tolerant GPIO (TT or FT) . . . 496
Figure 71. Input floating/pull-up/pull-down configurations . . . 500
Figure 72. Output configuration . . . 501
Figure 73. Alternate function configuration . . . 502
Figure 74. High-impedance analog configuration . . . 502
Figure 75. SBS block diagram . . . 515
Figure 76. SBS boot control . . . 516
Figure 77. Compensation cell management . . . 519
Figure 78. Compensation cell use . . . 519
Figure 79. HPDMA block diagram . . . 557
Figure 80. HPDMA channel direct programming without linked-list (HPDMA_CxLLR = 0) . . . 559
Figure 81. HPDMA channel suspend and resume sequence . . . 560
Figure 82. HPDMA channel abort and restart sequence . . . 561
Figure 83. HPDMA channel terminate and drain FIFO sequence . . . 562
Figure 84. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x . . . 564
Figure 85. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x . . . 565
Figure 86. HPDMA dynamic linked-list data structure of linear addressing channel x . . . 566
Figure 87. HPDMA dynamic linked-list data structure of a 2D addressing channel x . . . 566
Figure 88. HPDMA channel execution and linked-list programming in run-to-completion mode (HPDMA_CxCR.LSM = 0) . . . 568
Figure 89. Inserting a LLIn with an auxiliary HPDMA channel y . . . 570
Figure 90. HPDMA channel execution and linked-list programming in link step mode (HPDMA_CxCR.LSM = 1) . . . 572
Figure 91. Building LLIn+1: HPDMA dynamic linked-lists in link step mode . . . 573
Figure 92. Replace with a new LLIn' in register file in link step mode . . . 574
Figure 93. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 1) . . . 575
Figure 94. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 2) . . . 576
Figure 95. HPDMA channel execution and linked-list programming . . . 578
Figure 96. Programmed 2D addressing . . . 581

Figure 97. HPDMA arbitration policy . . . 596
Figure 98. Trigger hit, memorization and overrun waveform . . . 600
Figure 99. HPDMA circular buffer programming: update of the memory start address with a linear addressing channel . . . 601
Figure 100. Shared HPDMA channel with circular buffering: update of the memory start address with a linear addressing channel . . . 602
Figure 101. GPDMA block diagram . . . 648
Figure 102. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0) . . . 649
Figure 103. GPDMA channel suspend and resume sequence . . . 650
Figure 104. GPDMA channel abort and restart sequence . . . 651
Figure 105. GPDMA channel terminate and drain FIFO sequence . . . 652
Figure 106. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x . . . 654
Figure 107. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x . . . 655
Figure 108. GPDMA dynamic linked-list data structure of a linear addressing channel x . . . 656
Figure 109. GPDMA dynamic linked-list data structure of a 2D addressing channel x . . . 656
Figure 110. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0) . . . 658
Figure 111. Inserting a LLIn with an auxiliary GPDMA channel y . . . 660
Figure 112. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1) . . . 662
Figure 113. Building LLIn+1: GPDMA dynamic linked-lists in link step mode . . . 663
Figure 114. Replace with a new LLIn' in register file in link step mode . . . 664
Figure 115. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 1) . . . 665
Figure 116. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 2) . . . 666
Figure 117. GPDMA channel execution and linked-list programming . . . 668
Figure 118. Programmed 2D addressing . . . 671
Figure 119. GPDMA arbitration policy . . . 678
Figure 120. Trigger hit, memorization, and overrun waveform . . . 681
Figure 121. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel . . . 682
Figure 122. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel . . . 683
Figure 123. Stream processing engine . . . 718
Figure 124. Chaining of processing units . . . 719
Figure 125. Neural-ART 11 NPU accelerator . . . 720
Figure 126. Stream engines . . . 722
Figure 127. Two dimensional lossy compression . . . 723
Figure 128. Decompression unit (DECUN) . . . 724
Figure 129. Convolutional accelerator (CONVACC) . . . 725
Figure 130. Example of pooling operations . . . 726
Figure 131. Pooling unit (POOL) . . . 726
Figure 132. Activation unit (ACTIV) . . . 727
Figure 133. Arithmetic unit (ARITH) . . . 730
Figure 134. Memory to memory transfer . . . 731
Figure 135. Simple processing . . . 732
Figure 136. Multiple processing . . . 732
Figure 137. Classical Conv-Pool-ReLU . . . 732
Figure 138. EXTI block diagram . . . 745

Figure 139. Configurable event triggering logic CPU wake-up . . . 747
Figure 140. Direct event triggering logic CPU wake up . . . 748
Figure 141. CRC calculation unit block diagram . . . 767
Figure 142. CORDIC convergence for trigonometric functions . . . 781
Figure 143. CORDIC convergence for hyperbolic functions . . . 782
Figure 144. CORDIC convergence for square root . . . 783
Figure 145. Block diagram . . . 792
Figure 146. Input buffer areas . . . 794
Figure 147. Circular input buffer . . . 795
Figure 148. Circular input buffer operation . . . 796
Figure 149. Circular output buffer . . . 797
Figure 150. Circular output buffer operation . . . 798
Figure 151. FIR filter structure . . . 800
Figure 152. IIR filter structure (direct form 1) . . . 802
Figure 153. X1 buffer initialization . . . 807
Figure 154. Filtering example 1 . . . 808
Figure 155. Filtering example 2 . . . 809
Figure 156. HSP functional view . . . 820
Figure 157. HSP block diagram . . . 823
Figure 158. Single-precision floating-point format . . . 827
Figure 159. MMC overview . . . 828
Figure 160. Memory mapping seen from the AHB slave port SP1 . . . 830
Figure 161. BRAM arbitration . . . 831
Figure 162. HSP code loading . . . 832
Figure 163. HSP start-up . . . 834
Figure 164. MSGB block diagram . . . 835
Figure 165. DCMD block diagram . . . 838
Figure 166. Direct command programming example . . . 840
Figure 167. STREAM block diagram . . . 841
Figure 168. BUFCMB details . . . 842
Figure 169. Buffer usage generation example . . . 844
Figure 170. TRGITF block diagram . . . 846
Figure 171. TRGIN overrun detection . . . 848
Figure 172. SEG block diagram . . . 849
Figure 173. Example of dedicated event generation . . . 851
Figure 174. EVTC block diagram . . . 852
Figure 175. Processing request control . . . 853
Figure 176. Example of synchronization mechanism . . . 854
Figure 177. Task controller details . . . 857
Figure 178. Priority encoder example . . . 858
Figure 179. HSP interrupt controller . . . 859
Figure 180. HSP error interrupt controller . . . 860
Figure 181. HSP event interrupt controller . . . 864
Figure 182. Saturation flag . . . 865
Figure 183. HCC block diagram . . . 868
Figure 184. TRGITF wake-up capabilities . . . 869
Figure 185. TCU block diagram . . . 870
Figure 186. Task overlap . . . 870
Figure 187. TRGO block diagram . . . 871
Figure 188. TSC block diagram . . . 872
Figure 189. Trace controller block diagram . . . 874
Figure 190. TSC capture example 1 . . . 876

Figure 191. TSC capture example 2 . . . 877
Figure 192. SNOOP function . . . 879
Figure 193. Break interface . . . 881
Figure 194. Break input timing example . . . 883
Figure 195. Step-by-step timing example . . . 885
Figure 196. PLSUP simplified algorithm . . . 890
Figure 197. Processing list programming algorithm . . . 892
Figure 198. Direct command processing list simplified algorithm . . . 893
Figure 199. Data sharing models . . . 894
Figure 200. FMC block diagram . . . 960
Figure 201. FMC memory banks (default mapping) . . . 963
Figure 202. Mode 1 read access waveforms . . . 974
Figure 203. Mode 1 write access waveforms . . . 975
Figure 204. Mode A read access waveforms . . . 977
Figure 205. Mode A write access waveforms . . . 978
Figure 206. Mode 2 and mode B read access waveforms . . . 980
Figure 207. Mode 2 write access waveforms . . . 980
Figure 208. Mode B write access waveforms . . . 981
Figure 209. Mode C read access waveforms . . . 983
Figure 210. Mode C write access waveforms . . . 983
Figure 211. Mode D read access waveforms . . . 986
Figure 212. Mode D write access waveforms . . . 986
Figure 213. Muxed read access waveforms . . . 989
Figure 214. Muxed write access waveforms . . . 989
Figure 215. Asynchronous wait during a read access waveforms . . . 992
Figure 216. Asynchronous wait during a write access waveforms . . . 992
Figure 217. Wait configuration waveforms . . . 994
Figure 218. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM) . . . 995
Figure 219. Synchronous multiplexed write mode waveforms - PSRAM (CRAM) . . . 997
Figure 220. NAND flash controller waveforms for common memory access . . . 1011
Figure 221. Access to non ‘CE don’t care’ NAND-flash . . . 1012
Figure 222. Burst write SDRAM access waveforms . . . 1022
Figure 223. Burst read SDRAM access . . . 1023
Figure 224. Logic diagram of Read access with RBURST bit set (RPIPE=0) . . . 1024
Figure 225. Read access crossing row boundary . . . 1026
Figure 226. Write access crossing row boundary . . . 1026
Figure 227. Self-refresh mode . . . 1028
Figure 228. Power-down mode . . . 1029
Figure 229. XSPI block diagram for octal configuration . . . 1041
Figure 230. XSPI block diagram for quad configuration . . . 1041
Figure 231. XSPI block diagram for dual-quad configuration . . . 1042
Figure 232. SDR read command in octal configuration . . . 1044
Figure 233. DTR read in octal-SPI mode with DQS (Macronix mode) example . . . 1047
Figure 234. SDR write command in octal-SPI mode example . . . 1049
Figure 235. DTR write in octal-SPI mode (Macronix mode) example . . . 1049
Figure 236. Example of HyperBus read operation (8-bit data mode) . . . 1051
Figure 237. HyperBus write operation with initial latency . . . 1052
Figure 238. HyperBus read operation with additional latency . . . 1052
Figure 239. HyperBus write operation with additional latency . . . 1053
Figure 240. HyperBus write operation with no latency (register write) . . . 1053
Figure 241. HyperBus read operation page crossing with latency . . . 1054
Figure 242. D0/D1 data ordering in octal-SPI DTR mode (Micron) - Read access . . . 1060

Figure 243. D0/D1 data ordering in octal-SPI DTR mode with DQS and CLK polarity aligned during Read access . . . 1061
Figure 244. OctaRAM read operation with reverse data ordering D1/D0 . . . 1061
Figure 245. NCS when CKMODE = 0 (T = CLK period) . . . 1067
Figure 246. Example of software control of two external memories (octal mode) . . . 1068
Figure 247. Example of hardware-controller extended memory support (octal mode) . . . 1069
Figure 248. SDMMC “no response” and “no data” operations . . . 1098
Figure 249. SDMMC (multiple) block read operation . . . 1098
Figure 250. SDMMC (multiple) block write operation . . . 1099
Figure 251. SDMMC (sequential) stream read operation . . . 1099
Figure 252. SDMMC (sequential) stream write operation . . . 1099
Figure 253. SDMMC block diagram . . . 1101
Figure 254. SDMMC Command and data phase relation . . . 1103
Figure 255. Control unit . . . 1105
Figure 256. Command/response path . . . 1106
Figure 257. Command path state machine (CPSM) . . . 1107
Figure 258. Data path . . . 1113
Figure 259. DDR mode data packet clocking . . . 1114
Figure 260. DDR mode CRC status / boot acknowledgment clocking . . . 1114
Figure 261. Data path state machine (DPSM) . . . 1115
Figure 262. CLKMUX unit . . . 1126
Figure 263. Linked list structures . . . 1128
Figure 264. Asynchronous interrupt generation . . . 1131
Figure 265. Synchronous interrupt period data read . . . 1132
Figure 266. Synchronous interrupt period data write . . . 1132
Figure 267. Asynchronous interrupt period data read . . . 1133
Figure 268. Asynchronous interrupt period data write . . . 1134
Figure 269. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25 . . . 1137
Figure 270. Clock stop with SDMMC_CK for DDR50, SDR50, SDR104 . . . 1137
Figure 271. Read Wait with SDMMC_CK < 50 MHz . . . 1138
Figure 272. Read Wait with SDMMC_CK > 50 MHz . . . 1138
Figure 273. CMD12 stream timing . . . 1141
Figure 274. CMD5 Sleep Awake procedure . . . 1143
Figure 275. Normal boot mode operation . . . 1145
Figure 276. Alternative boot mode operation . . . 1146
Figure 277. Command response R1b busy signaling . . . 1147
Figure 278. SDMMC state control . . . 1148
Figure 279. Card cycle power / power up diagram . . . 1149
Figure 280. CMD11 signal voltage switch sequence . . . 1150
Figure 281. Voltage switch transceiver typical application . . . 1152
Figure 282. DLYB block diagram . . . 1180
Figure 283. ADC block diagram . . . 1188
Figure 284. ADC clock scheme . . . 1195
Figure 285. ADC1 connectivity . . . 1196
Figure 286. ADC2 connectivity . . . 1197
Figure 287. ADC3 connectivity . . . 1198
Figure 288. ADC4 connectivity . . . 1199
Figure 289. ADC calibration . . . 1202
Figure 290. Updating the ADC calibration factor . . . 1203
Figure 291. Mixing single-ended and differential channels . . . 1204
Figure 292. Enabling/disabling the ADC . . . 1205
Figure 293. Bulb mode timing diagram . . . 1208

Figure 294. Analog-to-digital conversion time . . . 1213
Figure 295. Stopping ongoing regular conversions . . . 1214
Figure 296. Stopping ongoing regular and injected conversions . . . 1214
Figure 297. Triggers shared between ADC master and slave . . . 1216
Figure 298. Injected conversion latency during ongoing regular conversion . . . 1217
Figure 299. Example of ADC_JSQR queue of context (sequence change) . . . 1219
Figure 300. Example of ADC_JSQR queue of context (trigger change) . . . 1219
Figure 301. Example of ADC_JSQR queue of context with overflow before conversion . . . 1220
Figure 302. Example of ADC_JSQR queue of context with overflow during conversion . . . 1220
Figure 303. Example of ADC_JSQR queue of context with empty queue (case JQM = 0) . . . 1221
Figure 304. Example of ADC_JSQR queue of context with empty queue (JQM = 1) . . . 1222
Figure 305. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion . . . 1222
Figure 306. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion and a new trigger occurs . . . 1223
Figure 307. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs outside an ongoing conversion . . . 1223
Figure 308. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 1) . . . 1224
Figure 309. Flushing ADC_JSQR queue of context by setting ADDIS = 1 (JQM = 0) . . . 1224
Figure 310. Flushing ADC_JSQR queue of context by setting ADDIS = 1 (JQM = 1) . . . 1225
Figure 311. Single conversions of a sequence, software trigger . . . 1227
Figure 312. Continuous conversion of a sequence, software trigger . . . 1227
Figure 313. Single conversions of a sequence, hardware trigger . . . 1228
Figure 314. Continuous conversions of a sequence, hardware trigger . . . 1228
Figure 315. Right alignment (offset disabled, unsigned value) . . . 1230
Figure 316. Right alignment (offset enabled, signed value) . . . 1231
Figure 317. Left alignment (offset disabled, unsigned value) . . . 1231
Figure 318. Left alignment (offset enabled, signed value) . . . 1232
Figure 319. Example of overrun (OVRMOD = 0) . . . 1235
Figure 320. Example of overrun (OVRMOD = 1) . . . 1235
Figure 321. AUTDLY = 1, regular conversion in continuous mode, software trigger . . . 1239
Figure 322. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0) . . . 1239
Figure 323. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1) . . . 1240
Figure 324. AUTDLY = 1, regular continuous conversions interrupted by injected conversions . . . 1240
Figure 325. AUTDLY = 1 in autoinjection mode (JAUTO = 1) . . . 1241
Figure 326. Analog watchdog guarded area . . . 1242
Figure 327. ADC_AWDy_OUT signal generation (on all regular channels) . . . 1244
Figure 328. ADC_AWDy_OUT signal generation (AWDy flag not cleared by software) . . . 1245
Figure 329. ADC_AWDy_OUT signal generation (on a single regular channel) . . . 1245
Figure 330. ADC_AWDy_OUT signal generation (on all injected channels) . . . 1246
Figure 331. 12-bit result oversampling with 10-bit right shift and rounding . . . 1247
Figure 332. Triggered regular oversampling mode (TROVS bit = 1) . . . 1249
Figure 333. Regular oversampling modes (4x ratio) . . . 1250
Figure 334. Regular and injected oversampling modes used simultaneously . . . 1251
Figure 335. Triggered regular oversampling with injection . . . 1251
Figure 336. Oversampling in autoinjection mode . . . 1252
Figure 337. Triggered injected parallel oversampling for 3 channels example . . . 1253
Figure 338. Injected oversampling with intermediate value enabled example . . . 1254
Figure 339. Triggered injected parallel oversampling for 3 channels with

accumulated data example . . . 1254
Figure 340. Dual ADC block diagram . . . 1256
Figure 341. Injected simultaneous mode on four channels: Dual ADC mode . . . 1257
Figure 342. Regular simultaneous mode on 16 channels: dual ADC mode . . . 1259
Figure 343. Interleaved mode on 1 channel in continuous conversion mode: Dual ADC mode . . . 1261
Figure 344. Interleaved mode on 1 channel in single conversion mode: Dual ADC mode . . . 1261
Figure 345. Interleaved conversion with injection . . . 1262
Figure 346. Alternate trigger: injected group of each ADC . . . 1263
Figure 347. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode . . . 1264
Figure 348. Alternate + regular simultaneous . . . 1264
Figure 349. Case of trigger occurring during injected conversion . . . 1265
Figure 350. Interleaved single channel CH0 with injected sequence CH11, CH12 . . . 1265
Figure 351. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 case 1: Master interrupted first . . . 1266
Figure 352. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 case 2: Slave interrupted first . . . 1266
Figure 353. DMA requests in regular simultaneous mode when DAMDF[1:0] = 0b00 . . . 1267
Figure 354. DMA requests in interleaved mode when DAMDF = 0b10 . . . 1268
Figure 355. Temperature sensor channel block diagram . . . 1271
Figure 356. VBAT channel block diagram . . . 1272
Figure 357. VREFINT channel block diagram . . . 1273
Figure 358. Temperature sensor functional block diagram . . . 1323
Figure 359. Method for low REF_CLK frequencies . . . 1325
Figure 360. Method for high REF_CLK frequencies . . . 1325
Figure 361. Temperature sensor sequence . . . 1328
Figure 362. Dual-channel DAC block diagram . . . 1341
Figure 363. Data registers in single DAC channel mode . . . 1345
Figure 364. Data registers in dual DAC channel mode . . . 1346
Figure 365. Timing diagram for conversion with trigger disabled TEN = 0 . . . 1347
Figure 366. DAC LFSR register calculation algorithm . . . 1349
Figure 367. DAC conversion (SW trigger enabled) with LFSR wave generation . . . 1350
Figure 368. DAC triangle wave generation . . . 1350
Figure 369. DAC conversion (SW trigger enabled) with triangle wave generation . . . 1351
Figure 370. DAC sawtooth wave generation (STDIRx = 0) . . . 1352
Figure 371. DAC sawtooth wave generation (STDIRx = 1) . . . 1352
Figure 372. DAC sawtooth STINCTRIG and STRSTTRIG priority (STDIR = 0) . . . 1353
Figure 373. DAC extended sawtooth wave generation (STDIRx = 0) . . . 1354
Figure 374. DAC extended sawtooth wave generation (STDIRx = 1) . . . 1355
Figure 375. DAC sample and hold mode phase diagram . . . 1358
Figure 376. VREFBUF block diagram . . . 1390
Figure 377. Comparator functional block diagram . . . 1398
Figure 378. Window comparator mode . . . 1402
Figure 379. Comparator hysteresis . . . 1403
Figure 380. Comparator output digital filter (N=4) . . . 1404
Figure 381. Comparator output digital filter (N=4, with spurious) . . . 1404
Figure 382. Comparator output blanking . . . 1405
Figure 383. COMP interrupt through NVIC circuitry . . . 1407
Figure 384. Scaler block diagram . . . 1408
Figure 385. ADF block diagram . . . 1416
Figure 386. SITF overview . . . 1418

Figure 387. SPI timing example . . . 1419
Figure 388. Manchester timing example (SITFMOD = 11) . . . 1420
Figure 389. CKGEN overview . . . 1423
Figure 390. BSMX overview . . . 1425
Figure 391. DFLT overview . . . 1426
Figure 392. Programmable delay . . . 1427
Figure 393. CIC4 and CIC5 frequency response with decimation ratio = 32 or 16 . . . 1428
Figure 394. Reshape filter frequency response normalized (FRS / 2 = 1) . . . 1434
Figure 395. Trigger logic for DFLT and CKGEN . . . 1436
Figure 396. Asynchronous continuous mode (ACQMOD[2:0] = 0) . . . 1437
Figure 397. Asynchronous single-shot mode (ACQMOD[2:0] = 001) . . . 1438
Figure 398. Synchronous continuous mode (ACQMOD[2:0] = 010) . . . 1439
Figure 399. Synchronous single-shot mode (ACQMOD[2:0] = 011) . . . 1440
Figure 400. Window continuous mode (ACQMOD[2:0] = 100) . . . 1441
Figure 401. Discard function example . . . 1443
Figure 402. Start sequence with DFLTEN, in continuous mode, audio configuration . . . 1443
Figure 403. SAD block diagram . . . 1444
Figure 404. SAD flow diagram . . . 1446
Figure 405. SAD timing diagram example . . . 1452
Figure 406. ADF_DFLTxDR data format . . . 1452
Figure 407. Data resynchronization . . . 1453
Figure 408. Example of overflow and transfer to memory . . . 1454
Figure 409. ADF interrupt interface . . . 1457
Figure 410. Sensor connection examples . . . 1462
Figure 411. Global frequency response . . . 1462
Figure 412. Detailed frequency response . . . 1463
Figure 413. Simplified DFLT view with gain information . . . 1465
Figure 414. SAD example working with SADMOD = 01 . . . 1468
Figure 415. SAD example working with SADMOD = 1x . . . 1469
Figure 416. RNG block diagram . . . 1489
Figure 417. NIST SP800-90B entropy source model . . . 1490
Figure 418. RNG initialization overview . . . 1493
Figure 419. HASH block diagram . . . 1509
Figure 420. Message data swapping feature . . . 1511
Figure 421. HASH suspend/resume mechanism . . . 1517
Figure 422. High-resolution timer overview . . . 1533
Figure 423. Timer A..F overview . . . 1543
Figure 424. Continuous timer operation . . . 1544
Figure 425. Single-shot timer operation . . . 1545
Figure 426. Timer reset resynchronization (prescaling ratio above 32) . . . 1546
Figure 427. Repetition rate versus HRTIM_REPxR content in continuous mode . . . 1547
Figure 428. Repetition counter behavior in single-shot mode . . . 1548
Figure 429. Compare events action on outputs: set on compare 1, reset on compare 2 . . . 1549
Figure 430. Timer A timing unit capture circuitry . . . 1553
Figure 431. Auto-delayed overview (compare 2 only) . . . 1554
Figure 432. Auto-delayed compare . . . 1555
Figure 433. Triggered-half mode example . . . 1557
Figure 434. Triggered-half mode timeout . . . 1558
Figure 435. Push-pull mode block diagram . . . 1559
Figure 436. Push-pull mode example . . . 1560
Figure 437. Push-Pull with deadtime . . . 1561
Figure 438. Deadtime generation . . . 1562

Figure 439. Complementary outputs with deadtime insertion 1562
Figure 440. Deadtime insertion versus deadtime sign (1 indicates negative deadtime) 1563
Figure 441. Complementary outputs for low pulsewidth (SDTRx = SDTFx = 0) 1564
Figure 442. Complementary outputs for low pulsewidth (SDTRx = SDTFx = 1) 1564
Figure 443. Complementary outputs for low pulsewidth (SDTRx = 0, SDTFx = 1) 1564
Figure 444. Complementary outputs for low pulsewidth (SDTRx = 1, SDTFx=0) 1565
Figure 445. Crossbar override mode 1566
Figure 446. Delayed fault operation 1568
Figure 447. Deadtime on Fault 1568
Figure 448. Crossbar override force on event mode 1569
Figure 449. Delayed Fault exit example 1570
Figure 450. Master timer overview 1571
Figure 451. Basic symmetric waveform in up-down counting mode 1572
Figure 452. Complex symmetric waveform in up-down counting mode 1573
Figure 453. Asymmetric waveform in up-down counting mode 1573
Figure 454. External event management in up-down counting mode 1574
Figure 455. Interleaved up-down counter operation 1574
Figure 456. Interleaved up-down counter operation 1575
Figure 457. Push-pull up-down mode example 1575
Figure 458. Up-down mode with “greater than” comparison 1576
Figure 459. Up-down mode with output set on period event, for OUTROM[1:0]=10 1577
Figure 460. Repetition counter behavior in up-down counting mode 1578
Figure 461. External event conditioning overview (1 channel represented) 1581
Figure 462. Latency to external events (counter reset and output set) 1585
Figure 463. Latency to external events (output reset on external event) 1585
Figure 464. Event blanking mode 1586
Figure 465. Event postpone mode 1586
Figure 466. External event blanking with edge-sensitive trigger 1587
Figure 467. External event blanking, level sensitive triggering 1588
Figure 468. Event windowing mode 1588
Figure 469. External event windowing with edge-sensitive trigger 1590
Figure 470. External event windowing, level sensitive triggering 1590
Figure 471. External event counters 1591
Figure 472. External event counter cumulative mode (EEVxRSTM = 1, EEVxCNT = 2) 1592
Figure 473. Delayed Idle mode entry 1593
Figure 474. Burst mode and delayed protection priorities (DIDL = 0) 1594
Figure 475. Burst mode and delayed protection priorities (DIDL = 1) 1595
Figure 476. Balanced Idle protection example 1596
Figure 477. Resynchronized timer update (TAU=1 in HRTIM_TIMBCR) 1601
Figure 478. Early turn-ON and early turn-OFF behavior in “greater than” PWM mode 1603
Figure 479. Output management overview 1608
Figure 480. HRTIM output states and transitions 1609
Figure 481. Burst mode operation example 1610
Figure 482. Burst mode trigger on external event 1613
Figure 483. Delayed burst mode entry with deadtime enabled and IDLESx = 1 1614
Figure 484. Delayed burst mode entry during deadtime 1615
Figure 485. Burst mode exit when the deadtime generator is enabled 1616
Figure 486. Burst mode emulation example 1618
Figure 487. Carrier frequency signal insertion 1618
Figure 488. HRTIM outputs with Chopper mode enabled 1619
Figure 489. Fault protection circuitry (FAULT1 fully represented, FAULT2..8 partially) 1620
Figure 490. Fault signal filtering (FLTxF[3:0]= 0010: $f_{SAMPLING}$ = fHRTIM, N = 4) 1621

Figure 491. Fault counter cumulative mode (FLTxRSTM = 1, FLTxCNT[3:0] = 2) 1623
Figure 492. Auxiliary outputs 1625
Figure 493. Auxiliary and main outputs during burst mode (DIDLx = 0) 1626
Figure 494. Deadtime distortion on auxiliary output when exiting burst mode. 1626
Figure 495. Counter behavior in synchronized start mode 1630
Figure 496. ADC trigger selection overview 1631
Figure 497. ADC triggers 1632
Figure 498. ADC trigger post-scaling in up-counting mode 1633
Figure 499. ADC trigger post-scaling in up/down counting mode 1633
Figure 500. Combining several updates on a single hrtim_dac_trgx output 1634
Figure 501. DAC triggers for slope compensation 1637
Figure 502. DAC triggers overview 1638
Figure 503. DMA burst overview 1642
Figure 504. Burst DMA operation flowchart 1643
Figure 505. Registers update following DMA burst transfer 1644
Figure 506. Buck converter topology 1646
Figure 507. Dual Buck converter management 1647
Figure 508. Synchronous rectification depending on output current 1647
Figure 509. Buck with synchronous rectification 1648
Figure 510. 3-phase interleaved buck converter 1648
Figure 511. 3-phase interleaved buck converter control 1649
Figure 512. Transition mode PFC 1650
Figure 513. Transition mode PFC waveforms 1651
Figure 514. Advanced-control timer block diagram 1828
Figure 515. Counter timing diagram with prescaler division change from 1 to 2 1836
Figure 516. Counter timing diagram with prescaler division change from 1 to 4 1837
Figure 517. Counter timing diagram, internal clock divided by 1 1838
Figure 518. Counter timing diagram, internal clock divided by 2 1839
Figure 519. Counter timing diagram, internal clock divided by 4 1839
Figure 520. Counter timing diagram, internal clock divided by N. 1840
Figure 521. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded). 1840
Figure 522. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded). 1841
Figure 523. Counter timing diagram, internal clock divided by 1 1842
Figure 524. Counter timing diagram, internal clock divided by 2 1843
Figure 525. Counter timing diagram, internal clock divided by 4 1843
Figure 526. Counter timing diagram, internal clock divided by N. 1844
Figure 527. Counter timing diagram, update event when repetition counter is not used. 1844
Figure 528. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6 1846
Figure 529. Counter timing diagram, internal clock divided by 2 1846
Figure 530. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36 1847
Figure 531. Counter timing diagram, internal clock divided by N. 1847
Figure 532. Counter timing diagram, update event with ARPE = 1 (counter underflow) 1848
Figure 533. Counter timing diagram, Update event with ARPE = 1 (counter overflow). 1849
Figure 534. Update rate examples depending on mode and TIM_RCR register settings 1850
Figure 535. External trigger input block 1851
Figure 536. Control circuit in normal mode, internal clock divided by 1. 1852
Figure 537. tim_ti2 external clock connection example 1852
Figure 538. Control circuit in external clock mode 1 1853
Figure 539. External trigger input block 1854
Figure 540. Control circuit in external clock mode 2 1855

Figure 541. Capture/compare channel (example: channel 1 input stage) . . . 1855
Figure 542. Capture/compare channel 1 main circuit . . . 1856
Figure 543. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4) . . . 1857
Figure 544. Output stage of capture/compare channel (channel 5, idem ch. 6 and 7) . . . 1857
Figure 545. PWM input mode timing . . . 1859
Figure 546. Output compare mode, toggle on tim_oc1 . . . 1861
Figure 547. Edge-aligned PWM waveforms (ARR = 8) . . . 1863
Figure 548. Center-aligned PWM waveforms (ARR = 8) . . . 1864
Figure 549. Dithering principle . . . 1865
Figure 550. Data format and register coding in dithering mode . . . 1866
Figure 551. PWM resolution vs frequency . . . 1867
Figure 552. PWM dithering pattern . . . 1868
Figure 553. Dithering effect on duty cycle in center-aligned PWM mode . . . 1869
Figure 554. Generation of 2 phase-shifted PWM signals with 50% duty cycle . . . 1870
Figure 555. Asymmetric mode example for single-shunt current sensing . . . 1873
Figure 556. Asymmetric mode 8 example . . . 1873
Figure 557. Combined PWM mode examples . . . 1875
Figure 558. Combined PWM 3 example . . . 1876
Figure 559. 3-phase combined PWM signals with multiple trigger pulses per period . . . 1877
Figure 560. Complementary output with symmetrical dead-time insertion . . . 1878
Figure 561. Asymmetrical deadtime . . . 1879
Figure 562. Dead-time waveforms with delay greater than the negative pulse . . . 1879
Figure 563. Dead-time waveforms with delay greater than the positive pulse . . . 1879
Figure 564. Clock prescalers for dead-time and digital filters . . . 1880
Figure 565. Break and Break2 circuitry overview . . . 1882
Figure 566. Various output behavior in response to a break event on tim_brk (OSSI = 1) . . . 1884
Figure 567. PWM output state following tim_brk and tim_brk2 assertion (OSSI = 1) . . . 1885
Figure 568. PWM output state following tim_brk assertion (OSSI = 0) . . . 1886
Figure 569. Immediate break and Delayed 1/2 break modes . . . 1886
Figure 570. Pulse termination during delayed break . . . 1887
Figure 571. Delayed break with deadtime insertion . . . 1888
Figure 572. Output redirection (tim_brk2 request not represented) . . . 1890
Figure 573. tim_ocref_clr input selection multiplexer . . . 1891
Figure 574. Clearing TIMx tim_ocxref . . . 1892
Figure 575. Asymmetric PWM mode 1 . . . 1892
Figure 576. Asymmetric PWM mode 2 . . . 1893
Figure 577. Asymmetric PWM mode 7 (OC1refc = oc1ref OR oc5ref) . . . 1893
Figure 578. Combined PWM mode 1 . . . 1894
Figure 579. 6-step generation, COM example (OSSR = 1) . . . 1895
Figure 580. Example of one pulse mode. . . . 1896
Figure 581. Retriggerable one-pulse mode . . . 1897
Figure 582. Pulse generator circuitry . . . 1898
Figure 583. Pulse generation on compare event, for edge-aligned and encoder modes . . . 1899
Figure 584. Extended pulsewidth in case of concurrent triggers . . . 1900
Figure 585. Example of counter operation in encoder interface mode. . . . 1902
Figure 586. Example of encoder interface mode with tim_ti1fp1 polarity inverted. . . . 1902
Figure 587. Quadrature encoder counting modes . . . 1903
Figure 588. Direction plus clock encoder mode . . . 1904
Figure 589. Directional clock encoder mode (CC1P = CC2P = 0) . . . 1904
Figure 590. Directional clock encoder mode (CC1P = CC2P = 1) . . . 1905
Figure 591. Index gating options . . . 1906
Figure 592. Jittered Index signals . . . 1906

Figure 593. Index generation for IPOS[1:0] = 11 . . . 1907
Figure 594. Counter reading with index gated on channel A (IPOS[1:0] = 11) . . . 1907
Figure 595. Counter reading with index ungated (IPOS[1:0] = 00) . . . 1908
Figure 596. Counter reading with index gated on channel A and B . . . 1908
Figure 597. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11) . . . 1909
Figure 598. Counter reset Narrow index pulse (closer view, ARR = 0x07) . . . 1910
Figure 599. Index behavior in x1 and x2 mode (IPOS[1:0] = 01) . . . 1911
Figure 600. Directional index sensitivity . . . 1911
Figure 601. Counter reset as function of FIDX bit setting . . . 1912
Figure 602. Index blanking . . . 1912
Figure 603. Index behavior in clock + direction mode, IPOS[0] = 1 . . . 1913
Figure 604. Index behavior in directional clock mode, IPOS[0] = 1 . . . 1913
Figure 605. State diagram for quadrature encoded signals . . . 1914
Figure 606. Up-counting encoder error detection . . . 1915
Figure 607. Down-counting encode error detection . . . 1916
Figure 608. Encoder mode change with preload transferred on update (SMSPS = 0) . . . 1917
Figure 609. Encoder with built-in debouncer, x2 mode . . . 1918
Figure 610. Encoder with built-in debouncer, x4 mode . . . 1918
Figure 611. Clock drift measure using directional clock mode . . . 1919
Figure 612. Measuring time interval between edges on three signals . . . 1920
Figure 613. XOR gate connection with TI1S = 1 and XORPS = 0 . . . 1921
Figure 614. XOR gate connection with TI1S = 1 and XORPS = 1 . . . 1921
Figure 615. Example of Hall sensor interface . . . 1923
Figure 616. Control circuit in reset mode . . . 1924
Figure 617. Control circuit in Gated mode . . . 1925
Figure 618. Control circuit in trigger mode . . . 1926
Figure 619. Control circuit in external clock mode 2 + trigger mode . . . 1927
Figure 620. tim_trgo2 postscaler example . . . 1928
Figure 621. General-purpose timer block diagram . . . 2008
Figure 622. Counter timing diagram with prescaler division change from 1 to 2 . . . 2015
Figure 623. Counter timing diagram with prescaler division change from 1 to 4 . . . 2016
Figure 624. Counter timing diagram, internal clock divided by 1 . . . 2017
Figure 625. Counter timing diagram, internal clock divided by 2 . . . 2017
Figure 626. Counter timing diagram, internal clock divided by 4 . . . 2018
Figure 627. Counter timing diagram, internal clock divided by N . . . 2018
Figure 628. Counter timing diagram, Update event when ARPE = 0 (TIM_ARR not preloaded) . . . 2019
Figure 629. Counter timing diagram, Update event when ARPE = 1 (TIM_ARR preloaded) . . . 2020
Figure 630. Counter timing diagram, internal clock divided by 1 . . . 2021
Figure 631. Counter timing diagram, internal clock divided by 2 . . . 2022
Figure 632. Counter timing diagram, internal clock divided by 4 . . . 2022
Figure 633. Counter timing diagram, internal clock divided by N . . . 2023
Figure 634. Counter timing diagram, Update event . . . 2023
Figure 635. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6 . . . 2025
Figure 636. Counter timing diagram, internal clock divided by 2 . . . 2025
Figure 637. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36 . . . 2026
Figure 638. Counter timing diagram, internal clock divided by N . . . 2026
Figure 639. Counter timing diagram, Update event with ARPE = 1 (counter underflow) . . . 2027
Figure 640. Counter timing diagram, Update event with ARPE = 1 (counter overflow) . . . 2028
Figure 641. External trigger input block . . . 2028
Figure 642. Control circuit in normal mode, internal clock divided by 1 . . . 2030
Figure 643. tim_ti2 external clock connection example . . . 2030
Figure 644. Control circuit in external clock mode 1 . . . 2031

Figure 645. External trigger input block . . . 2032
Figure 646. Control circuit in external clock mode 2 . . . 2033
Figure 647. Capture/compare channel (example: channel 1 input stage) . . . 2033
Figure 648. Capture/compare channel 1 main circuit . . . 2034
Figure 649. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4) . . . 2034
Figure 650. PWM input mode timing . . . 2037
Figure 651. Output compare mode, toggle on tim_oc1 . . . 2039
Figure 652. Edge-aligned PWM waveforms (ARR = 8) . . . 2040
Figure 653. Center-aligned PWM waveforms (ARR = 8) . . . 2041
Figure 654. Dithering principle . . . 2042
Figure 655. Data format and register coding in dithering mode . . . 2043
Figure 656. PWM resolution vs frequency (16-bit mode) . . . 2044
Figure 657. PWM resolution vs frequency (32-bit mode) . . . 2044
Figure 658. PWM dithering pattern . . . 2045
Figure 659. Dithering effect on duty cycle in center-aligned PWM mode . . . 2046
Figure 660. Generation of two phase-shifted PWM signals with 50% duty cycle . . . 2048
Figure 661. Combined PWM mode on channels 1 and 3 . . . 2050
Figure 662. Clock prescalers for dead-time and digital filters . . . 2051
Figure 663. OCREF_CLR input selection multiplexer . . . 2051
Figure 664. Clearing TIMx tim_ocxref . . . 2052
Figure 665. Asymmetric PWM mode 1 . . . 2053
Figure 666. Asymmetric PWM mode 2 . . . 2053
Figure 667. Combined PWM mode 1 . . . 2054
Figure 668. Example of one-pulse mode . . . 2055
Figure 669. Retriggerable one-pulse mode . . . 2056
Figure 670. Pulse generator circuitry . . . 2057
Figure 671. Pulse generation on compare event, for edge-aligned and encoder modes . . . 2058
Figure 672. Extended pulse width in case of concurrent triggers . . . 2059
Figure 673. Example of counter operation in encoder interface mode . . . 2061
Figure 674. Example of encoder interface mode with tim_ti1fp1 polarity inverted . . . 2061
Figure 675. Quadrature encoder counting modes . . . 2062
Figure 676. Direction plus clock encoder mode . . . 2063
Figure 677. Directional clock encoder mode (CC1P = CC2P = 0) . . . 2064
Figure 678. Directional clock encoder mode (CC1P = CC2P = 1) . . . 2064
Figure 679. Index gating options . . . 2066
Figure 680. Jittered Index signals . . . 2066
Figure 681. Index generation for IPOS[1:0] = 11 . . . 2067
Figure 682. Counter reading with index gated on channel A (IPOS[1:0] = 11) . . . 2067
Figure 683. Counter reading with index ungated (IPOS[1:0] = 00) . . . 2068
Figure 684. Counter reading with index gated on channel A and B . . . 2068
Figure 685. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11) . . . 2069
Figure 686. Counter reset Narrow index pulse (closer view, ARR = 0x07) . . . 2070
Figure 687. Index behavior in x1 and x2 mode (IPOS[1:0] = 01) . . . 2071
Figure 688. Directional index sensitivity . . . 2071
Figure 689. Counter reset as function of FIDX bit setting . . . 2072
Figure 690. Index blanking . . . 2072
Figure 691. Index behavior in clock + direction mode, IPOS[0] = 1 . . . 2073
Figure 692. Index behavior in directional clock mode, IPOS[0] = 1 . . . 2073
Figure 693. State diagram for quadrature encoded signals . . . 2074
Figure 694. Up-counting encoder error detection . . . 2075
Figure 695. Down-counting encode error detection . . . 2076
Figure 696. Encoder mode change with preload transferred on update (SMSPS = 0) . . . 2077

Figure 697. Encoder with built-in debouncer, x2 mode . . . 2078
Figure 698. Encoder with built-in debouncer, x4 mode . . . 2078
Figure 699. Clock drift measure using directional clock mode . . . 2079
Figure 700. Measuring time interval between edges on three signals . . . 2080
Figure 701. XOR gate connection with TI1S = 1 and XORPS = 0 . . . 2081
Figure 702. XOR gate connection with TI1S = 1 and XORPS = 1 . . . 2081
Figure 703. Control circuit in reset mode . . . 2082
Figure 704. Control circuit in gated mode . . . 2083
Figure 705. Control circuit in trigger mode . . . 2084
Figure 706. Control circuit in external clock mode 2 + trigger mode . . . 2085
Figure 707. Master/Slave timer example . . . 2086
Figure 708. Master/slave connection example with 1 channel only timers . . . 2086
Figure 709. Gating TIM_slv with tim_oc1ref of TIM_mstr . . . 2087
Figure 710. Gating TIM_slv with Enable of TIM_mstr . . . 2088
Figure 711. Triggering TIM_slv with update of TIM_mstr . . . 2089
Figure 712. Triggering TIM_slv with Enable of TIM_mstr . . . 2089
Figure 713. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input . . . 2090
Figure 714. Basic timer block diagram . . . 2135
Figure 715. Control circuit in normal mode, internal clock divided by 1 . . . 2136
Figure 716. Counter timing diagram with prescaler division change from 1 to 2 . . . 2137
Figure 717. Counter timing diagram with prescaler division change from 1 to 4 . . . 2138
Figure 718. Counter timing diagram, internal clock divided by 1 . . . 2139
Figure 719. Counter timing diagram, internal clock divided by 2 . . . 2139
Figure 720. Counter timing diagram, internal clock divided by 4 . . . 2140
Figure 721. Counter timing diagram, internal clock divided by N . . . 2140
Figure 722. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded) . . . 2141
Figure 723. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded) . . . 2142
Figure 724. Dithering principle . . . 2143
Figure 725. Data format and register coding in dithering mode . . . 2143
Figure 726. FCnt resolution vs frequency . . . 2144
Figure 727. PWM dithering pattern . . . 2144
Figure 728. General-purpose timer block diagram (TIM9/TIM12) . . . 2158
Figure 729. General-purpose timer block diagram (TIM10/TIM11/TIM13/TIM14) . . . 2159
Figure 730. Counter timing diagram with prescaler division change from 1 to 2 . . . 2163
Figure 731. Counter timing diagram with prescaler division change from 1 to 4 . . . 2163
Figure 732. Counter timing diagram, internal clock divided by 1 . . . 2164
Figure 733. Counter timing diagram, internal clock divided by 2 . . . 2165
Figure 734. Counter timing diagram, internal clock divided by 4 . . . 2165
Figure 735. Counter timing diagram, internal clock divided by N . . . 2166
Figure 736. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded) . . . 2166
Figure 737. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded) . . . 2167
Figure 738. Control circuit in normal mode, internal clock divided by 1 . . . 2168
Figure 739. tim_ti2 external clock connection example . . . 2168
Figure 740. Control circuit in external clock mode 1 . . . 2169
Figure 741. Capture/compare channel 1 input stage (TIM10/TIM11/TIM13/TIM14) . . . 2170
Figure 742. Capture/compare channel 1 input stage (TIM9/TIM12) . . . 2170
Figure 743. Capture/compare channel 1 main circuit . . . 2171
Figure 744. Output stage of capture/compare channel 1 . . . 2171

Figure 745. PWM input mode timing . . . 2173
Figure 746. Output compare mode, toggle on tim_oc1. . . . 2175
Figure 747. Edge-aligned PWM waveforms (ARR = 8) . . . 2176
Figure 748. Dithering principle . . . 2177
Figure 749. Data format and register coding in dithering mode . . . 2177
Figure 750. PWM resolution vs frequency . . . 2178
Figure 751. PWM dithering pattern . . . 2179
Figure 752. Combined PWM mode on channel 1 and 2 . . . 2181
Figure 753. Clock prescalers for digital filters . . . 2182
Figure 754. Example of one pulse mode . . . 2182
Figure 755. Retriggerable one pulse mode . . . 2184
Figure 756. Measuring time interval between edges on 2 signals . . . 2185
Figure 757. XOR gate connection with TI1S = 1 and XORPS = 0 . . . 2185
Figure 758. XOR gate connection with TI1S = 1 and XORPS = 1 . . . 2186
Figure 759. Control circuit in reset mode . . . 2187
Figure 760. Control circuit in gated mode . . . 2188
Figure 761. Control circuit in trigger mode . . . 2188
Figure 762. TIM15 block diagram . . . 2225
Figure 763. TIM16/TIM17 block diagram . . . 2226
Figure 764. Counter timing diagram with prescaler division change from 1 to 2 . . . 2232
Figure 765. Counter timing diagram with prescaler division change from 1 to 4 . . . 2232
Figure 766. Counter timing diagram, internal clock divided by 1 . . . 2234
Figure 767. Counter timing diagram, internal clock divided by 2 . . . 2234
Figure 768. Counter timing diagram, internal clock divided by 4 . . . 2235
Figure 769. Counter timing diagram, internal clock divided by N . . . 2235
Figure 770. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded) . . . 2236
Figure 771. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded) . . . 2237
Figure 772. Counter timing diagram, internal clock divided by 1 . . . 2238
Figure 773. Counter timing diagram, internal clock divided by 2 . . . 2239
Figure 774. Counter timing diagram, internal clock divided by 4 . . . 2239
Figure 775. Counter timing diagram, internal clock divided by N . . . 2240
Figure 776. Counter timing diagram, update event when repetition counter is not used . . . 2240
Figure 777. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6 . . . 2242
Figure 778. Counter timing diagram, internal clock divided by 2 . . . 2242
Figure 779. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36 . . . 2243
Figure 780. Counter timing diagram, internal clock divided by N . . . 2243
Figure 781. Counter timing diagram, update event with ARPE = 1 (counter underflow) . . . 2244
Figure 782. Counter timing diagram, Update event with ARPE = 1 (counter overflow) . . . 2245
Figure 783. Update rate examples depending on mode and TIM_RCR register settings . . . 2246
Figure 784. Control circuit in normal mode, internal clock divided by 1 . . . 2247
Figure 785. tim_ti2 external clock connection example . . . 2248
Figure 786. Control circuit in external clock mode 1 . . . 2249
Figure 787. Capture/compare channel (example: channel 1 input stage) . . . 2249
Figure 788. Capture/compare channel 1 main circuit . . . 2250
Figure 789. Output stage of capture/compare channel (channel 1) . . . 2250
Figure 790. Output stage of capture/compare channel (channel 2 for TIM15) . . . 2251
Figure 791. PWM input mode timing . . . 2253
Figure 792. Output compare mode, toggle on tim_oc1 . . . 2255
Figure 793. Edge-aligned PWM waveforms (ARR = 8) . . . 2256
Figure 794. Center-aligned PWM waveforms (ARR = 8) . . . 2257

Figure 795. Dithering principle . . . 2258
Figure 796. Data format and register coding in dithering mode . . . 2259
Figure 797. PWM resolution vs frequency . . . 2260
Figure 798. PWM dithering pattern . . . 2260
Figure 799. Dithering effect on duty cycle in center-aligned PWM mode . . . 2261
Figure 800. Generation of a phase-shifted PWM signal with 50% duty cycle . . . 2262
Figure 801. Combined PWM mode on channel 1 and 2 . . . 2264
Figure 802. Complementary output with symmetrical dead-time insertion. . . . 2265
Figure 803. Asymmetrical deadtime . . . 2266
Figure 804. Dead-time waveforms with delay greater than the negative pulse. . . . 2266
Figure 805. Dead-time waveforms with delay greater than the positive pulse. . . . 2266
Figure 806. Clock prescalers for dead-time and digital filters . . . 2267
Figure 807. Break circuitry overview . . . 2269
Figure 808. Output behavior in response to a break event on tim_brk . . . 2271
Figure 809. Immediate break and Delayed 1/2 break modes . . . 2272
Figure 810. Pulse termination during delayed break . . . 2273
Figure 811. Delayed break with deadtime insertion . . . 2273
Figure 812. Output redirection . . . 2275
Figure 813. tim_ocref_clr input selection multiplexer . . . 2276
Figure 814. Asymmetric PWM mode 1 . . . 2277
Figure 815. Asymmetric PWM mode 2 . . . 2277
Figure 816. Combined PWM mode 1 . . . 2278
Figure 817. 6-step generation, COM example (OSSR = 1) . . . 2279
Figure 818. Example of one pulse mode . . . 2280
Figure 819. Retriggerable one pulse mode . . . 2282
Figure 820. Example of counter operation in encoder interface mode. . . . 2284
Figure 821. Example of encoder interface mode with tim_ti1fp1 polarity inverted. . . . 2284
Figure 822. Quadrature encoder counting modes . . . 2285
Figure 823. Direction plus clock encoder mode . . . 2286
Figure 824. Directional clock encoder mode (CC1P = CC2P = 0) . . . 2286
Figure 825. Directional clock encoder mode (CC1P = CC2P = 1) . . . 2287
Figure 826. State diagram for quadrature encoded signals . . . 2288
Figure 827. Encoder mode change with preload transferred on update . . . 2288
Figure 828. Encoder with built-in debouncer, x2 mode . . . 2290
Figure 829. Encoder with built-in debouncer, x4 mode . . . 2290
Figure 830. Clock drift measure using directional clock mode . . . 2291
Figure 831. Measuring time interval between edges on 2 signals . . . 2292
Figure 832. XOR gate connection with TI1S = 1 and XORPS = 0 . . . 2292
Figure 833. XOR gate connection with TI1S = 1 and XORPS = 1 . . . 2293
Figure 834. Control circuit in reset mode . . . 2294
Figure 835. Control circuit in gated mode . . . 2295
Figure 836. Control circuit in trigger mode . . . 2296
Figure 837. LPTIM1/2/3 block diagram(1) . . . 2375
Figure 838. LPTIM4/5 block diagram(1) . . . 2376
Figure 839. Glitch filter timing diagram . . . 2380
Figure 840. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1) . . . 2382
Figure 841. LPTIM output waveform, single-counting mode configuration and Set-once mode activated (WAVE bit is set) . . . 2383
Figure 842. LPTIM output waveform, Continuous counting mode configuration . . . 2383
Figure 843. Waveform generation . . . 2385
Figure 844. Encoder mode counting sequence . . . 2389

Figure 845. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1). . . . 2390
Figure 846. Capture/compare input stage (channel 1) . . . 2391
Figure 847. Capture/compare output stage (channel 1) . . . 2391
Figure 848. Edge-aligned PWM mode (PRELOAD = 1) . . . 2393
Figure 849. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0) . . . 2394
Figure 850. PWM mode with immediate update versus preloaded update . . . 2395
Figure 851. Independent watchdog block diagram . . . 2425
Figure 852. Reset timing due to timeout . . . 2427
Figure 853. Reset timing due to refresh in the not allowed area . . . 2428
Figure 854. Changing PR, RL, and performing a refresh(1) . . . 2429
Figure 855. Window comparator update(1) . . . 2430
Figure 856. Independent watchdog interrupt timing diagram . . . 2432
Figure 857. Early wake-up comparator update(1) . . . 2433
Figure 858. Watchdog block diagram . . . 2441
Figure 859. Window watchdog timing diagram . . . 2443
Figure 860. RTC block diagram . . . 2448
Figure 861. TAMP block diagram . . . 2495
Figure 862. Backup registers protection zones . . . 2501
Figure 863. Tamper sampling with precharge pulse . . . 2506
Figure 864. Low level detection with precharge and filtering . . . 2506
Figure 865. Active tamper filtering . . . 2509
Figure 866. Block diagram . . . 2538
Figure 867. I²C-bus protocol . . . 2540
Figure 868. Setup and hold timings . . . 2542
Figure 869. I2C initialization flow . . . 2544
Figure 870. Data reception . . . 2545
Figure 871. Data transmission . . . 2546
Figure 872. Target initialization flow . . . 2549
Figure 873. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0 . . . 2551
Figure 874. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1 . . . 2552
Figure 875. Transfer bus diagrams for I2C target transmitter (mandatory events only) . . . 2553
Figure 876. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0 . . . 2554
Figure 877. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1 . . . 2555
Figure 878. Transfer bus diagrams for I2C target receiver (mandatory events only) . . . 2555
Figure 879. Controller clock generation . . . 2557
Figure 880. Controller initialization flow . . . 2559
Figure 881. 10-bit address read access with HEAD10R = 0 . . . 2559
Figure 882. 10-bit address read access with HEAD10R = 1 . . . 2560
Figure 883. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes . . . 2561
Figure 884. Transfer sequence flow for I2C controller transmitter, N > 255 bytes . . . 2562
Figure 885. Transfer bus diagrams for I2C controller transmitter (mandatory events only) . . . 2563
Figure 886. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes . . . 2565
Figure 887. Transfer sequence flow for I2C controller receiver, N > 255 bytes . . . 2566
Figure 888. Transfer bus diagrams for I2C controller receiver (mandatory events only) . . . 2567
Figure 889. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$ . . . 2571
Figure 890. Transfer sequence flow for SMBus target transmitter N bytes + PEC . . . 2575
Figure 891. Transfer bus diagram for SMBus target transmitter (SBC = 1) . . . 2575
Figure 892. Transfer sequence flow for SMBus target receiver N bytes + PEC . . . 2577

Figure 893. Bus transfer diagrams for SMBus target receiver (SBC = 1) 2578
Figure 894. Bus transfer diagrams for SMBus controller transmitter 2579
Figure 895. Bus transfer diagrams for SMBus controller receiver 2581
Figure 896. I3C block diagram 2608
Figure 897. I3C (primary) controller state and programming sequence diagram. 2613
Figure 898. I3C target state and programing sequence diagram, when initialized on an I3C bus without a static address 2618
Figure 899. I3C target state and programing sequence diagram when initialized on an I3C bus with a static address, ... 2619
Figure 900. I3C target state and programing sequence diagram, when initialized on an I2C bus with a static address 2620
Figure 901. I3C CCC messages, as controller 2634
Figure 902. I3C broadcast ENTDAA CCC, as controller 2635
Figure 903. I3C broadcast, direct read and direct write RSTACT CCC, as controller 2636
Figure 904. I3C CCC messages, as target 2638
Figure 905. I3C broadcast ENTDAA CCC, as target 2639
Figure 906. I3C broadcast DEFTGTS CCC, as target 2640
Figure 907. I3C broadcast DEFGRPA CCC, as target 2641
Figure 908. I3C private read/write messages, as controller. 2643
Figure 909. I3C private read/write messages, as target 2644
Figure 910. Legacy I2C read/write messages, as controller 2645
Figure 911. Legacy I^2C read/write messages, as target 2646
Figure 912. IBI transfer, as controller/target 2647
Figure 913. Hot-join request transfer, as controller/target 2649
Figure 914. Controller-role request transfer, as controller/target 2650
Figure 915. C-FIFO management, as controller 2651
Figure 916. TX-FIFO management, as controller 2653
Figure 917. RX-FIFO management, as controller 2655
Figure 918. S-FIFO management, as controller 2658
Figure 919. RX-FIFO management, as target on the I3C bus 2660
Figure 920. RX-FIFO management, as target on the I2C bus 2661
Figure 921. TX-FIFO management with I3C_TGTTDR, as target on the I3C bus 2663
Figure 922. TX-FIFO management with I3C_TGTTDR, as target on the I2C bus 2664
Figure 923. TX-FIFO management by software without I3C_TGTTDR if reading less bytes than TX-FIFO size, as target 2666
Figure 924. USART block diagram 2726
Figure 925. Word length programming 2730
Figure 926. Configurable stop bits 2732
Figure 927. TC/TXE behavior when transmitting 2734
Figure 928. Start bit detection when oversampling by 16 or 8 2735
Figure 929. usart_ker_ck clock divider block diagram 2738
Figure 930. Data sampling when oversampling by 16 2739
Figure 931. Data sampling when oversampling by 8 2740
Figure 932. Mute mode using Idle line detection 2747
Figure 933. Mute mode using address mark detection 2748
Figure 934. Break detection in LIN mode (11-bit break length - LBDL bit is set) 2751
Figure 935. Break detection in LIN mode vs. Framing error detection. 2752
Figure 936. USART example of synchronous master transmission. 2753
Figure 937. USART data clock timing diagram in synchronous master mode (M bits = 00) 2753
Figure 938. USART data clock timing diagram in synchronous master mode (M bits = 01) 2754

Figure 939. USART data clock timing diagram in synchronous slave mode (M bits = 00) . . . 2755
Figure 940. ISO 7816-3 asynchronous protocol . . . 2757
Figure 941. Parity error detection using the 1.5 stop bits . . . 2759
Figure 942. IrDA SIR ENDEC block diagram . . . 2763
Figure 943. IrDA data modulation (3/16) - normal mode . . . 2763
Figure 944. Transmission using DMA . . . 2765
Figure 945. Reception using DMA . . . 2766
Figure 946. Hardware flow control between two USARTs . . . 2766
Figure 947. RS232 RTS flow control . . . 2767
Figure 948. RS232 CTS flow control . . . 2768
Figure 949. Wake-up event verified (wake-up event = address match, FIFO disabled) . . . 2771
Figure 950. Wake-up event not verified (wake-up event = address match, FIFO disabled) . . . 2771
Figure 951. LPUART block diagram . . . 2816
Figure 952. LPUART word length programming . . . 2819
Figure 953. Configurable stop bits . . . 2821
Figure 954. TC/TXE behavior when transmitting . . . 2823
Figure 955. lpuart_ker_ck clock divider block diagram . . . 2827
Figure 956. Mute mode using Idle line detection . . . 2831
Figure 957. Mute mode using address mark detection . . . 2832
Figure 958. Transmission using DMA . . . 2834
Figure 959. Reception using DMA . . . 2835
Figure 960. Hardware flow control between two LPUARTs . . . 2836
Figure 961. RS232 RTS flow control . . . 2836
Figure 962. RS232 CTS flow control . . . 2837
Figure 963. Wake-up event verified (wake-up event = address match, FIFO disabled) . . . 2840
Figure 964. Wake-up event not verified (wake-up event = address match, FIFO disabled) . . . 2840
Figure 965. SPI/I2S block diagram . . . 2872
Figure 966. Full-duplex single master/ single slave application . . . 2875
Figure 967. Half-duplex single master/ single slave application . . . 2876
Figure 968. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode) . . . 2877
Figure 969. Master and three independent slaves connected in star topology . . . 2878
Figure 970. Master and three slaves connected in circular (daisy chain) topology . . . 2880
Figure 971. Multimaster application . . . 2881
Figure 972. Scheme of NSS control logic . . . 2883
Figure 973. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0) . . . 2883
Figure 974. NSS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0) . . . 2884
Figure 975. Data clock timing diagram . . . 2886
Figure 976. Data alignment when data size is not equal to 8, 16 or 32 bits . . . 2887
Figure 977. TI mode transfer . . . 2896
Figure 978. Optional configurations of the slave behavior when an underrun condition is detected . 2898
Figure 979. Waveform examples . . . 2906
Figure 980. Master I2S Philips protocol waveforms (16/32-bit full accuracy) . . . 2907
Figure 981. I2S Philips standard waveforms . . . 2907
Figure 982. Master MSB-justified 16- or 32-bit full-accuracy length . . . 2908
Figure 983. Master MSB-justified 16- or 24-bit data length . . . 2908
Figure 984. Slave MSB-justified 16-, 24- or 32-bit data length . . . 2909
Figure 985. LSB-justified 16 or 24-bit data length . . . 2909

Figure 986. Master PCM when the frame length is equal the data length . . . 2910
Figure 987. Master PCM standard waveforms (16 or 24-bit data length) . . . 2910
Figure 988. Slave PCM waveforms . . . 2911
Figure 989. Startup sequence, I2S Philips standard, master . . . 2914
Figure 990. Startup sequence, I2S Philips standard, slave . . . 2915
Figure 991. Stop sequence, I2S Philips standard, master . . . 2916
Figure 992. I^2S clock generator architecture . . . 2916
Figure 993. Data Format . . . 2919
Figure 994. Handling of underrun situation . . . 2921
Figure 995. Handling of underrun situation . . . 2921
Figure 996. Handling of overrun situation . . . 2922
Figure 997. Handling of overrun situation . . . 2922
Figure 998. Frame error detection, with FIXCH = 0 . . . 2923
Figure 999. Frame error detection, with FIXCH = 1 . . . 2924
Figure 1000. CAN subsystem . . . 2949
Figure 1001. FDCAN block diagram . . . 2951
Figure 1002. Bit timing . . . 2953
Figure 1003. Transceiver delay measurement . . . 2958
Figure 1004. Pin control in bus monitoring mode . . . 2959
Figure 1005. Pin control in loop-back mode . . . 2962
Figure 1006. CAN error state diagram . . . 2963
Figure 1007. Message RAM configuration . . . 2964
Figure 1008. Standard message ID filter path . . . 2967
Figure 1009. Extended message ID filter path . . . 2968
Figure 1010. OTG_FS full-speed block diagram . . . 3018
Figure 1011. OTG_FS A-B device connection . . . 3020
Figure 1012. OTG_FS peripheral-only connection . . . 3021
Figure 1013. OTG_FS host-only connection . . . 3025
Figure 1014. SOF connectivity (SOF trigger output to TIM and ITR1 connection) . . . 3029
Figure 1015. Updating OTG_HFIR dynamically (RLDCTRL = 1) . . . 3031
Figure 1016. Device-mode FIFO address mapping and AHB FIFO access mapping . . . 3032
Figure 1017. Host-mode FIFO address mapping and AHB FIFO access mapping . . . 3033
Figure 1018. Interrupt hierarchy . . . 3037
Figure 1019. Transmit FIFO write task . . . 3130
Figure 1020. Receive FIFO read task . . . 3131
Figure 1021. Normal bulk/control OUT/SETUP . . . 3132
Figure 1022. Bulk/control IN transactions . . . 3136
Figure 1023. Normal interrupt OUT . . . 3139
Figure 1024. Normal interrupt IN . . . 3144
Figure 1025. Isochronous OUT transactions . . . 3146
Figure 1026. Isochronous IN transactions . . . 3149
Figure 1027. Receive FIFO packet read . . . 3153
Figure 1028. Processing a SETUP packet . . . 3155
Figure 1029. Bulk OUT transaction . . . 3162
Figure 1030. TRDT max timing case . . . 3172
Figure 1031. UCPD block diagram . . . 3176
Figure 1032. Clock division and timing elements . . . 3177
Figure 1033. K-code transmission . . . 3180
Figure 1034. Transmit order for various sizes of data . . . 3181
Figure 1035. Packet format . . . 3182
Figure 1036. Line format of Hard Reset . . . 3182
Figure 1037. Line format of Cable Reset . . . 3183

Figure 1038. BIST test data frame . . . 3184
Figure 1039. BIST Carrier Mode 2 frame . . . 3184
Figure 1040. UCPD BMC transmitter architecture . . . 3185
Figure 1041. UCPD BMC receiver architecture . . . 3186
Figure 1042. Block diagram of debug infrastructure . . . 3213
Figure 1043. Clock domains of debug infrastructure . . . 3214
Figure 1044. SWD successful data transfer . . . 3218
Figure 1045. JTAG TAP state machine . . . 3219
Figure 1046. Debug and access port connections . . . 3231
Figure 1047. APB-D CoreSight component topology . . . 3242
Figure 1048. Embedded cross trigger . . . 3248
Figure 1049. Mapping of trigger inputs to outputs . . . 3251
Figure 1050. ETF state transition diagram . . . 3279
Figure 1051. Cortex-M7 CoreSight Topology . . . 3360

1 Documentation conventions

1.1 General information

The STM32H7Pxxx devices have an Arm® Cortex®-M7 with FPU core.

Note: Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

The Arm word and logo are trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved.

1.2 List of abbreviations for registers

The following abbreviations[(a)] are used in register descriptions:

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

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

1.3 Register reset value

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

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

1.4 Glossary

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

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

1.5 Availability of peripherals

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

1.6 Availability of security features

For security feature availability refer to the table below:

Table 1. Availability of security features

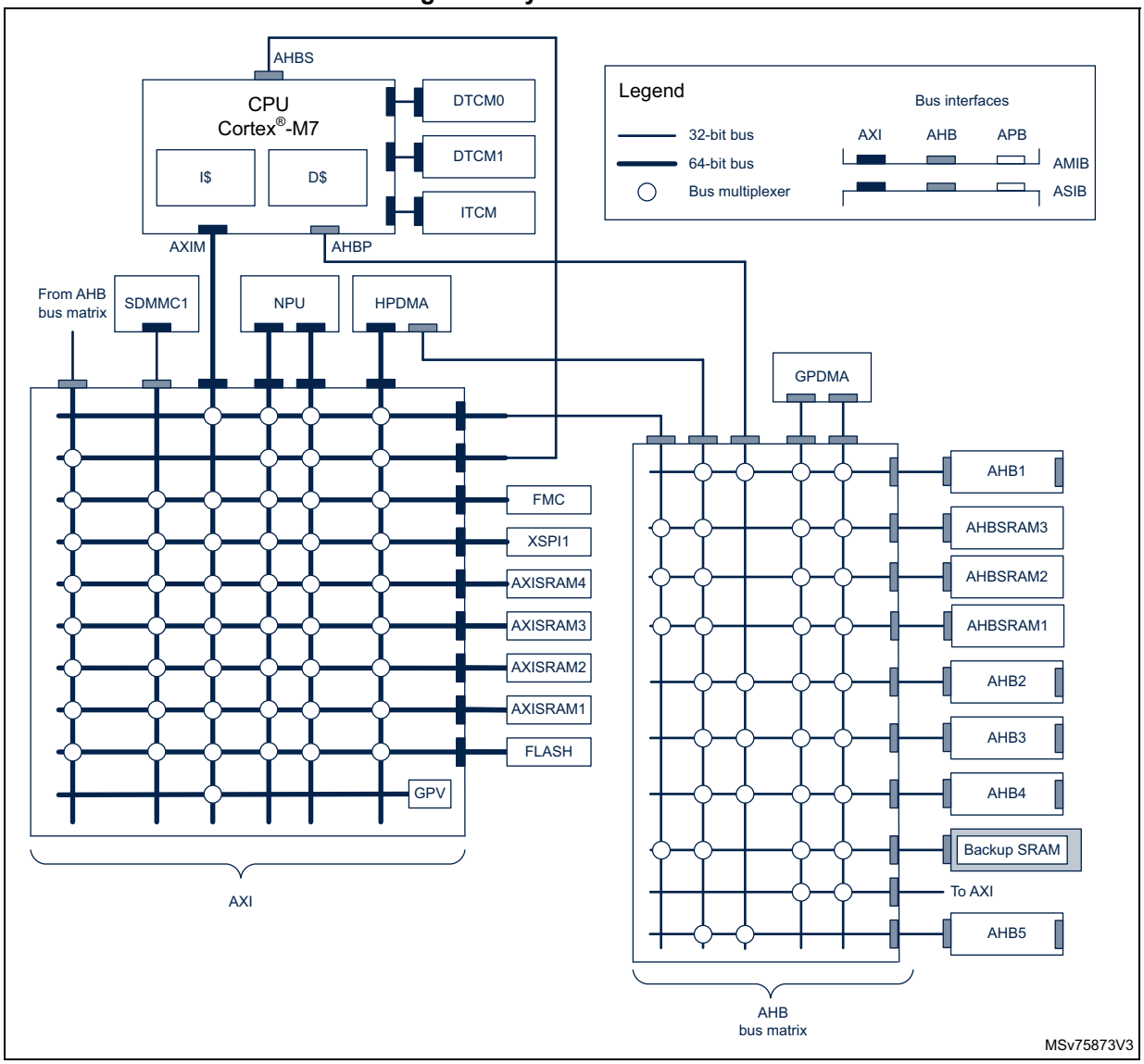
Security feature	STM32H7Pxxx
Embedded flash memory (FLASH): – Flash secure-only area	Available
Security memory management: – Root secure services (RSS)	
On-the-fly encryption / decryption	Not available
STiRoT	
Cryptographic processor (CRYP and PKA)	
Hash processor (HASH) with PKA verification always available	Available

2 Memory and bus architecture

2.1 System architecture

The STM32H7Pxxx architecture is illustrated in *Figure 1*. It is built around an Arm® Cortex®-M7 processing core optimized for execution by means of cache and tightly coupled memory (TCM). The processor accesses the system memory and peripherals through several bus interfaces.

Figure 1. System architecture



2.1.1 Bus matrices

The architecture features a 64-bit AMBA® AXI interconnect and a 32-bit multilayer AMBA AHB bus matrix, allowing interconnection of bus managers and subordinates. *Table 2* shows which managers can access which subordinates.

Table 2. Bus-manager-to-bus-subordinate interconnect

Bus subordinate / type(1)	Bus manager / type(1)								
	Cortex-M7 - AXIM	Cortex-M7 - AHBP	Cortex-M7 - ITCM	Cortex-M7 - DTCM	SDMMC1	HPDMA - AXI	HPDMA - AHB	GPDMA	NPU
	Interconnect path(2)								
ITCM	-	-	X	-	-	X	-	X	X
DTCM	-	-	-	X	-	X	-	X	X
FLASH	X	-	-	-	X	X	-	X	X
AXI SRAM1	X	-	-	-	X	X	-	X	X
AXI SRAM2	X	-	-	-	X	X	-	X	X
AXI SRAM3	X	-	-	-	X	X	-	X	X
AXI SRAM4	X	-	-	-	X	X	-	X	X
XSPI1	X	-	-	-	X	X	-	X	X
FMC	X	-	-	-	X	X	-	X	X
AHB SRAM1	X	-	-	-	-	X	X	X	X
AHB SRAM2	X	-	-	-	-	X	X	X	X
AHB SRAM3	X	-	-	-	-	X	X	X	X
AHB1 peripherals	-	X	-	-	-	-	X	X	-
APB1 peripherals	-	X	-	-	-	-	X	X	-
AHB2 peripherals	-	X	-	-	-	-	X	X	-
APB2 peripherals	-	X	-	-	-	-	X	X	-
AHB3 peripherals	-	X	-	-	-	-	X	X	-
AHB4 peripherals	-	X	-	-	-	-	X	X	-
APB4 peripherals	-	X	-	-	-	-	X	X	-
AHB5 peripherals	-	X	-	-	-	-	X	-	-
Backup RAM	X	-	-	-	-	X	X	X	-

1. Grey cells denote 64-bit bus, white cells denotes 32-bit bus.
2. “X” = access possible, “-” = access not possible.

AXI bus matrix

The AXI bus matrix arbitrates concurrent accesses from multiple managers to multiple subordinates. This allows efficient simultaneous operation of high-speed peripherals.

The arbitration uses a round-robin algorithm with QoS capability.

Refer to *Section 2.2: AXI interconnect matrix (AXIM)* for more information on AXI interconnect.

AHB bus matrix

The AHB bus matrix also arbitrates concurrent accesses from multiple managers to multiple subordinates. The arbitration uses a round-robin algorithm.

2.1.2 Peripheral buses

Peripheral functional units are accessed via 32-bit AMBA AHB and APB buses.

The APB1, APB2, and APB4 buses are connected to AHB1, AHB2, and AHB4 respectively, via AHB/APB bridges. These bridges provide semisynchronous interfacing, which allows the APB buses to operate at the same or a fraction of the AHB frequency that they connect to.

The AHB/APB bridges also allow APB1, APB2, and APB4 peripherals to connect to the GPDMA peripheral port, without going through the AHB bus matrix.

The AXI interconnect incorporates AHB/AXI bus bridges in subordinate bus interfaces (ASIBs) where required. External AXI/AHB bus bridges are used to connect AXI manager interfaces (AMIBs) to 32-bit AHB buses.

2.1.3 CPU buses

Cortex-M7 AXIM bus

The Cortex-M7 CPU uses the 64-bit AXIM bus to access all memories except ITCM and DTCM, including external memories on the FMC and XSPI interfaces.

The AXIM bus connects the CPU to the AXI interconnect.

Cortex-M7 ITCM bus

The Cortex-M7 CPU uses the 64-bit ITCM bus for fetching instructions from and accessing data in the ITCM.

Cortex-M7 DTCM bus

The Cortex-M7 CPU uses the dual 32-bit DTCM bus for accessing data in the DTCM. The dual DTCM bus allows load/load and load/store instruction pairs to be issued in parallel to the DTCM memory.

Cortex-M7 AHBP bus

The Cortex-M7 CPU uses the 32-bit AHBP bus for accessing AHB and APB peripherals via the AHB bus matrix.

Cortex-M7 AHBS bus

The Cortex-M7 CPU has a 32-bit AHBS subordinate port, which allows other bus managers (HPDMA controller, GPDMA controller, NPU) to access the ITCM and the DTCM.

2.1.4 Bus manager peripherals

SDMMC1

The SDMMC1 uses a 32-bit bus, connected to the AXI interconnect, through which it can access AXI SRAM and flash memories, and external memories through the XSPI and the FMC.

HPDMA-bus

The HPDMA controller has two bus managers: an AXI 64-bit bus, connected to the AXI interconnect, and an AHB 32-bit bus connected to the AHB bus matrix.

Through its AXI port, the HPDMA has access to all the memories in the system, including the TCM via the CPU AHBS port. Through its AHB port, it has access to all AHB and APB peripherals.

GPDMA-bus

The GPDMA controller has two 32-bit AHB bus managers, denoted M (memory) and P (peripheral). Both are connected to the AHB interconnect.

The managers allow GPDMA data transfers between two peripherals, between two memories or between a peripheral and a memory. Through each bus manager, the GPDMA controller can access all internal memories, external memories through the XSPI and the FMC, and all AHB and APB peripherals. The peripheral bus has a fast path to access the APB peripherals, bypassing the AHB bus matrix.

2.2 AXI interconnect matrix (AXIM)

2.2.1 AXI introduction

The AXI (advanced extensible interface) interconnect is based on the Arm® CoreLink™ NIC-400 network interconnect. The interconnect has six initiator ports, or ASIBs (AMBA subordinate interface blocks), and nine target ports, or AMIBs (AMBA manager interface blocks). The ASIBs are connected to the AMIBs via an AXI switch matrix.

Each ASIB is a subordinate on an AXI bus or AHB (advanced high-performance bus). Similarly, each AMIB is a manager on an AXI bus. Where an ASIB is connected to an AHB, it converts between the AHB and the AXI protocol.

The AXI interconnect includes a GPV (global programmer view) which contains registers for configuring certain parameters, such as the QoS (quality of service) level at each ASIB.

Any accesses to unallocated address space are handled by the default subordinate, which generates the return signals. This ensures that such transactions are completed and do not block the issuing manager and ASIB.

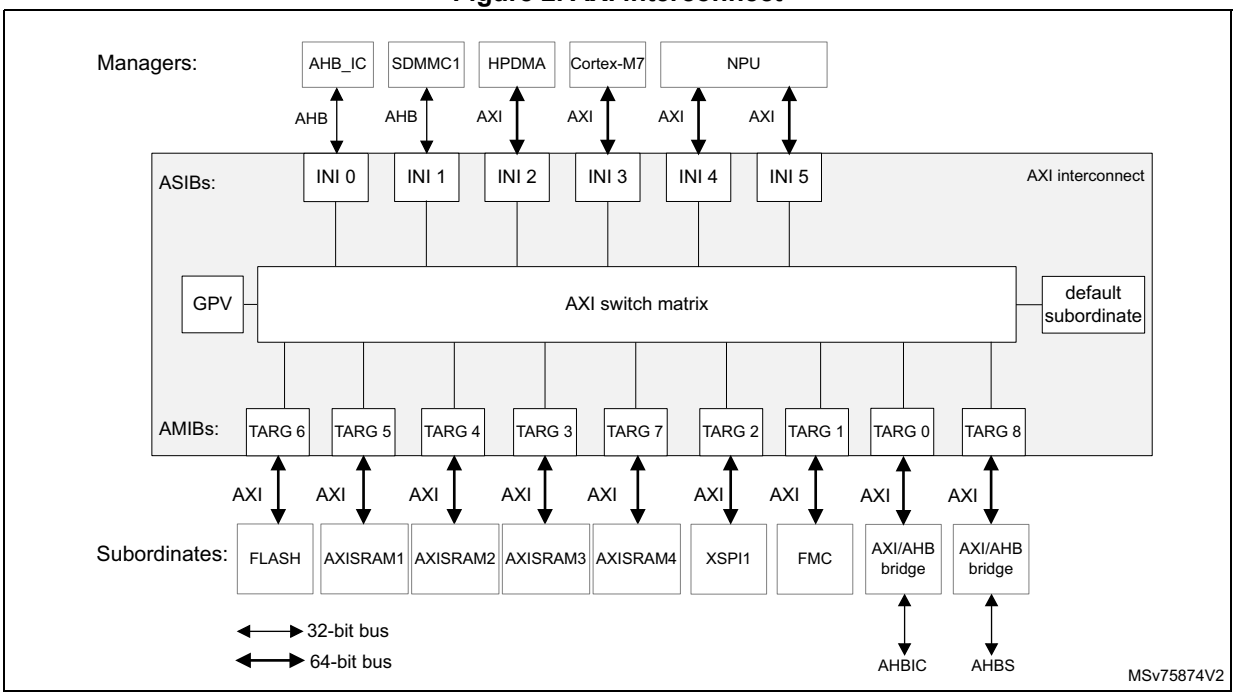
2.2.2 AXI interconnect main features

- 64-bit AXI bus switch matrix with six ASIBs and nine AMIBs
- AHB/AXI bridge function built into the ASIBs
- Concurrent connectivity of multiple ASIBs to multiple AMIBs
- Programmable traffic priority management (QoS - quality of service)
- Software-configurable via GPV

2.2.3 Block diagram

The AXI interconnect is shown in *Figure 2*.

Figure 2. AXI interconnect



ASIB configuration

Table 3 summarizes the characteristics of the ASIBs.

Table 3. ASIB configuration

ASIB	Connected manager	Protocol	Bus width	R/W issuing
INI 0	GPDMA(1)	AHB-lite	32	1/1
INI 1	SDMMC1	AHB-lite	32	1/1
INI 2	HPDMA	AXI4	64	8/8
INI 3	Cortex-M7	AXI4	64	7/32
INI 4	NPU	AXI4	64	16/16
INI 5		AXI4	64	16/16

1. Via AHB interconnect.

AMIB configuration

Table 4 summarizes the characteristics of the AMIBs.

Table 4. AMIB configuration

AMIB	Connected subordinate	Protocol	Bus width	R/W/Total acceptance
TARG 6	FLASH	AXI4	64	2/2/4
TARG 5	AXISRAM1	AXI4	64	2/2/4
TARG 4	AXISRAM2	AXI4	64	2/2/4
TARG 3	AXISRAM3	AXI4	64	2/2/4
TARG 7	AXISRAM4	AXI4	64	2/2/4
TARG 2	XSPI1	AXI4	64	2/2/4
TARG 1	FMC	AXI4	64	1/1/2
TARG 0	AHBIC	AXI4(1)	64	1/1/2
TARG 8	AHBS	AXI4(1)	64	1/1/2

1. The 64-bit AXI is converted to 32-bit AHB by an AXI/AHB bridge.

Quality of service (QoS)

The AXI switch matrix uses a priority-based arbitration when two ASIB simultaneously attempt to access the same AMIB. Each ASIB has programmable read channel and write channel priorities, known as QoS, from 0 to 15, such that the higher the value, the higher the priority. The read channel QoS value is programmed in the *AXI interconnect - INI x read QoS register (AXI_INIx_READ_QOS)*, and the write channel in the *AXI interconnect - INI x write QoS register (AXI_INIx_WRITE_QOS)*. The default QoS value for all channels is 0 (lowest priority).

If two coincident transactions arrive at the same AMIB, the higher priority transaction passes before the lower priority. If the two transactions have the same QoS value, then a least-recently-used (LRU) priority scheme is adopted.

The QoS values should be programmed according to the latency requirements for the application. Setting a higher priority for an ASIB ensures a lower latency for transactions initiated by the associated bus manager. This can be useful for real-time-constrained tasks. Assigning a high priority to managers (such as the Cortex-M7 CPU) that can make many and frequent accesses to the same subordinate can block access to that subordinate by other lower-priority managers.

Global programmer view (GPV)

The GPV contains configuration registers for the AXI interconnect. These registers are only accessible by the Cortex-M7 CPU.

2.2.4 AXI interconnect registers

AXI interconnect - peripheral ID4 register (AXI_PERIPH_ID_4)

Address offset: 0x1FD0

Reset value: 0x0000 0004

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COUNT4K[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **COUNT4K[3:0]**: Register file size
0x0: N/A

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm®

AXI interconnect - peripheral ID0 register (AXI_PERIPH_ID_0)

Address offset: 0x1FE0

Reset value: 0x0000 0000

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

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

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

Bits 7:0 **PARTNUM[7:0]**: Peripheral part number bits 0 to 7
0x00: Part number = 0x400

AXI interconnect - peripheral ID1 register (AXI_PERIPH_ID_1)

Address offset: 0x1FE4

Reset value: 0x0000 00B4

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity bits 0 to 3
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Peripheral part number bits 8 to 11
0x4: Part number = 0x400

AXI interconnect - peripheral ID2 register (AXI_PERIPH_ID_2)

Address offset: 0x1FE8

Reset value: 0x0000 006B

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

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

Bits 7:4 **REVISION[3:0]**: Peripheral revision number
0x6: r1p2

Bit 3 **JEDEC:** JEP106 code flag
0x1: JEDEC allocated code

Bits 2:0 **JEP106ID[6:4]**: JEP106 Identity bits 4 to 6
0x3: Arm® JEDEC code

AXI interconnect - component ID0 register (AXI_COMP_ID_0)

Address offset: 0x1FF0

Reset value: 0x0000 000D

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

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

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

Bits 7:0 **PREAMBLE[7:0]**: Preamble bits 0 to 7
0x0D: Common ID value

AXI interconnect - component ID1 register (AXI_COMP_ID_1)

Address offset: 0x1FF4

Reset value: 0x0000 00F0

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

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

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

Bits 7:4 **CLASS[3:0]**: Component class
0xF: Generic IP component class

Bits 3:0 **PREAMBLE[11:8]**: Preamble bits 8 to 11
0x0: Common ID value

AXI interconnect - component ID2 register (AXI_COMP_ID_2)

Address offset: 0x1FF8

Reset value: 0x0000 0005

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[19:12]**: Preamble bits 12 to 19
0x05: Common ID value

AXI interconnect - component ID3 register (AXI_COMP_ID_3)

Address offset: 0x1FFC

Reset value: 0x0000 00B1

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[27:20]**: Preamble bits 20 to 27
0xB1: Common ID value

AXI interconnect - TARG x bus matrix issuing functionality register (AXI_TARGx_FN_MOD_BM_ISS)

Address offset: 0x2008 + 0x1000 * x, (x = 0 to 8)

Reset value: 0x0000 0000

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

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

Bit 1 **WRITE_ISS_OVERRIDE**: Switch matrix write issuing override for target
- 0: Normal issuing capability
- 1: Set switch matrix write issuing capability to 1

Bit 0 **READ_ISS_OVERRIDE**: Switch matrix read issuing override for target
- 0: Normal issuing capability
- 1: Set switch matrix read issuing capability to 1

AXI interconnect - TARG x issuing functionality modification register (AXI_TARGx_FN_MOD)

Address offset: 0x2108+ 0x1000 * x, (x = 0 to 8)

Reset value: 0x0000 0000

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

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

Bit 1 **WRITE_ISS_OVERRIDE**: Override AMIB write issuing capability
- 0: Normal issuing capability
- 1: Force issuing capability to 1

Bit 0 **READ_ISS_OVERRIDE**: Override AMIB read issuing capability
- 0: Normal issuing capability
- 1: Force issuing capability to 1

AXI interconnect - TARG x bus matrix functionality 2 register (AXI_TARGx_FN_MOD2)

Address offset: 0x2024 + 0x1000 * x, (x = 0, 8)

Reset value: 0x0000 0000

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

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

Bit 0 **BYPASS_MERGE**: Disable packing of beats to match the output data width. Unaligned transactions are not realigned to the input data word boundary.

0: Normal operation

1: Disable packing

AXI interconnect - TARG x long burst functionality modification register (AXI_TARGx_FN_MOD_LB)

Address offset: 0x202C + 0x1000 * x, (x = 0, 8)

Reset value: 0x0000 0000

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

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

Bit 0 **FN_MOD_LB**: Controls burst breaking of long bursts

0: Long bursts can not be generated at the output of the AMIB

1: Long bursts can be generated at the output of the AMIB

AXI interconnect - INI x issuing functionality modification register (AXI_INIx_FN_MOD)

Address offset: 0x42108+ 0x1000 * x, (x = 0 to 5)

Reset value: 0x0000 0000

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

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

Bit 1 **WRITE_ISS_OVERRIDE**: Override ASIB write issuing capability
0: Normal issuing capability
1: Force issuing capability to 1

Bit 0 **READ_ISS_OVERRIDE**: Override ASIB read issuing capability
0: Normal issuing capability
1: Force issuing capability to 1

AXI interconnect - INI x read QoS register (AXI_INIx_READ_QOS)

Address offset: 0x42100 + 0x1000 * x, (x = 0 to 5)

Reset value: 0x0000 0006 (x = 0)

Reset value: 0x0000 0004 (x = 1)

Reset value: 0x0000 0008 (x = 2)

Reset value: 0x0000 0000 (x = 3)

Reset value: 0x0000 0002 (x = 4)

Reset value: 0x0000 0002 (x = 5)

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

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

Bits 3:0 **RQOS[3:0]**: Read channel QoS setting
0x0: Lowest priority
0xF: Highest priority

AXI interconnect - INI x write QoS register (AXI_INIx_WRITE_QOS)

Address offset: 0x42104 + 0x1000 * x, (x = 0 to 5)

Reset value: 0x0000 0006 (x = 0)

Reset value: 0x0000 0004 (x = 1)

Reset value: 0x0000 0008 (x = 2)

Reset value: 0x0000 0000 (x = 3)

Reset value: 0x0000 0002 (x = 4)

Reset value: 0x0000 0002 (x = 5)

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

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

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

Bits 3:0 **WQOS[3:0]**: Write channel QoS setting
0x0: Lowest priority
0xF: Highest priority

AXI interconnect - INI x functionality modification register 2 (AXI_INIx_FN_MOD2)

Address offset: 0x42024 + 0x1000 * x, (x = 0, 1)

Reset value: 0x0000 0000

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

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

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

Bit 0 **BYPASS_MERGE**: Disable packing of beats to match the output data width. Unaligned transactions are not realigned to the input data word boundary.
0: Normal operation
1: Disable packing

AXI interconnect - INI x functionality modification AHB register (AXI_INIx_FN_MOD_AHB)

Address offset: 0x42028 + 0x1000 * x, (x = 0, 1)

Reset value: 0x0000 0000

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

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

Bit 1 **WR_INCR_OVERRIDE**: Converts all AHB-Lite write transactions to a series of single beat AXI transactions, and each AHB-Lite write beat is acknowledged with the AXI buffered write response.

0: Normal operation

1: Write INCR override

Bit 0 **RD_INCR_OVERRIDE**: Converts all AHB-Lite read transactions to a series of single beat AXI transactions.

0: Normal operation

1: Read INCR override

2.2.5 AXI interconnect register map

Table 5. AXI interconnect register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1FD0	AXI_PERIPH_ID_4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0x1FD4	AXI_PERIPH_ID_5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Reserved							
	Reset value																									0	0	0	0	0	0	0	0
0x1FD8	AXI_PERIPH_ID_6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Reserved							
	Reset value																									0	0	0	0	0	0	0	0
0x1FDC	AXI_PERIPH_ID_7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Reserved							
	Reset value																									0	0	0	0	0	0	0	0

Table 5. AXI interconnect register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1FE0	AXI_PERIPH_ID_0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x1FE4	AXI_PERIPH_ID_1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	0	1	0	0
0x1FE8	AXI_PERIPH_ID_2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	1	1	0	1	0	1	1
0x1FEC	AXI_PERIPH_ID_3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Reserved							
	Reset value																									0	0	0	0	0	0	0	0
0x1FF0	AXI_COMP_ID_0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0x1FF4	AXI_COMP_ID_1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	1	0	0	0	0
0x1FF8	AXI_COMP_ID_2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0x1FFC	AXI_COMP_ID_3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1
0x2000 - 0x2004	Reserved																																
0x2008	AXI_TARG0_FN_MOD_BM_ISS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x200C - 0x2020	Reserved																																

Table 5. AXI interconnect register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2024	AXI_TARG0_ FN_MOD2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
	Reset value																																0
0x2028	Reserved																																
0x202C	AXI_TARG0_ FN_MOD_LB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FN_MOD_LB
	Reset value																																0
0x2030 - 0x2104	Reserved																																
0x2108	AXI_TARG0_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x210C - 0x3004	Reserved																																
0x3008	AXI_TARG1_ FN_MOD_BM_ ISS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x300C - 0x3104	Reserved																																
0x3108	AXI_TARG1_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x310C - 0x4004	Reserved																																

Table 5. AXI interconnect register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4008	AXI_TARG2_ FN_MOD_ BM_ISS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x400C - 0x4104	Reserved																																
0x4108	AXI_TARG2_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x410C - 0x5004	Reserved																																
0x5008	AXI_TARG3_ FN_MOD_BM_ ISS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x500C - 0x5104	Reserved																																
0x5108	AXI_TARG3_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x510C - 0x6004	Reserved																																

Table 5. AXI interconnect register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x6008	AXI_TARG4_ FN_MOD_BM_ ISS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x600C - 0x6104	Reserved																																
0x6108	AXI_TARG4_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x610C - 0x7004	Reserved																																
0x7008	AXI_TARG5_ FN_MOD_BM_ ISS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x700C - 0x7104	Reserved																																
0x7108	AXI_TARG5_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x710C - 0x8004	Reserved																																

Table 5. AXI interconnect register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x8008	AXI_TARG6_ FN_MOD_BM_ ISS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x800C - 0x8104	Reserved																																
0x8108	AXI_TARG6_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x810C- 0x9004	Reserved																																
0x9008	AXI_TARG7_ FN_MOD_BM_ ISS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x900C - 0x9104	Reserved																																
0x9108	AXI_TARG7_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x910C - 0xA004	Reserved																																

Table 5. AXI interconnect register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA008	AXI_TARG8_ FN_MOD_BM_ ISS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0xA00C - 0xA020	Reserved																																
0xA024	AXI_TARG8_ FN_MOD2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
	Reset value																																0
0xA028	Reserved																																
0xA02C	AXI_TARG8_ FN_MOD_LB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FN_MOD_LB
	Reset value																																0
0xA030 - 0xA104	Reserved																																
0xA108	AXI_TARG8_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0xA10C- 0x42020	Reserved																																
0x42024	AXI_INI0_ FN_MOD2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
	Reset value																																0

Table 5. AXI interconnect register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x42028	AXI_INI0_FN_MOD_AHB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_INC_OVERRIDE	RD_INC_OVERRIDE
	Reset value																															0	0
0x4202C-0x420FC	Reserved																																
0x42100	AXI_INI0_READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS [3:0]			
	Reset value																													0	1	1	0
0x42104	AXI_INI0_WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS [3:0]			
	Reset value																													0	1	1	0
0x42108	AXI_INI0_FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4210C-0x43020	Reserved																																
0x43024	AXI_INI1_FN_MOD2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
	Reset value																																0
0x43028	AXI_INI1_FN_MOD_AHB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_INC_OVERRIDE	RD_INC_OVERRIDE
	Reset value																															0	0
0x4302C-0x430FC	Reserved																																
0x43100	AXI_INI1_READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS [3:0]			
	Reset value																													0	1	0	0

Table 5. AXI interconnect register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43104	AXI_INI1_ WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS [3:0]			
	Reset value																													0	1	0	0
0x43108	AXI_INI1_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4310C - 0x440FC	Reserved																																
0x44100	AXI_INI2_ READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS [3:0]			
	Reset value																													1	0	0	0
0x44104	AXI_INI2_ WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS [3:0]			
	Reset value																													1	0	0	0
0x44108	AXI_INI2_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4410C- 0x450FC	Reserved																																
0x45100	AXI_INI3_ READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS [3:0]			
	Reset value																													0	0	0	0
0x45104	AXI_INI3_ WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS [3:0]			
	Reset value																													0	0	0	0
0x45108	AXI_INI3_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4510C- 0x460FC	Reserved																																

Table 5. AXI interconnect register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x46100	AXI_INI4_ READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS [3:0]			
	Reset value																													0	0	1	0
0x46104	AXI_INI4_ WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS [3:0]			
	Reset value																													0	0	1	0
0x46108	AXI_INI4_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4610C-0x470FC	Reserved																																
0x47100	AXI_INI5_ READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS [3:0]			
	Reset value																													0	0	1	0
0x47104	AXI_INI5_ WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS [3:0]			
	Reset value																													0	0	1	0
0x47108	AXI_INI5_ FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

2.3 Memory organization

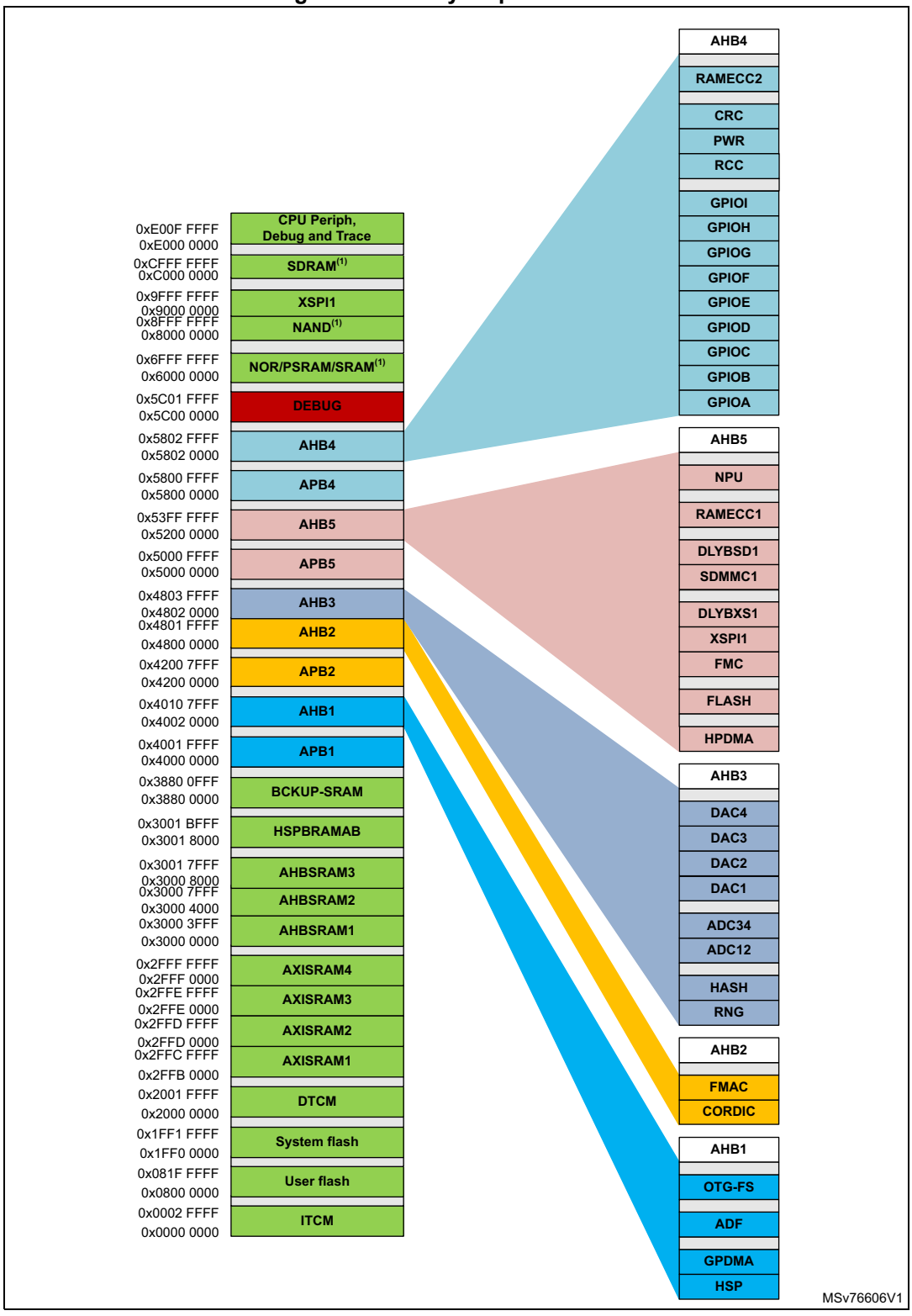
2.3.1 Introduction

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

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

2.3.2 Memory map and register boundary addresses

Figure 3. Memory map



1. External memory accessed via the FMC.

All the memory map areas that are not allocated to on-chip memories and peripheral units are considered “Reserved”.

The default memory attribute and the boundary addresses of the available peripheral units are shown in *Table 6* and *Table 7*.

Table 6. Memory map and default device memory area attribute

Region	Boundary address	Description	Type	Attributes	Execute never
External Devices	0xD0000000 - 0xDFFFFFFF	Reserved	Device	-	Yes
	0xC0000000 - 0xCFFFFFFF	FMC SDRAM or FMC NOR/PSRAM/SRAM remap			
	0xBFF00000 - 0xBFFFFFFF	GPV/AXIM			
	0xA0000000 - 0xBFF00000	Reserved			
External Memories	0x90000000 - 0x9FFFFFFF	XSPI1	Normal	Write-through cache attribute	No
	0x80000000 - 0x8FFFFFFF	FMC NAND flash memory			
	0x70000000 - 0x7FFFFFFF	Reserved		Write-back, write allocate cache attribute	
	0x60000000 - 0x6FFFFFFF	FMC NOR/PSRAM/SRAM or FMC SDRAM remap			
Peripherals	0x40000000 - 0x5FFFFFFF	Peripherals. See *Table 7: Peripheral boundary addresses*.	Device	-	Yes
RAM	0x38801000 - 0x3FFFFFFF	Reserved	Normal	Write-back, write allocate cache attribute	No
	0x38800000 - 0x38800FFF	BKPSRAM			
	0x3001C000 - 0x387FFFFF	Reserved			
	0x38018000 - 0x3001BFFF	HSPBRAMAB			
	0x30008000 - 0x30017FFF	AHBSRAM3			
	0x30004000 - 0x30007FFF	AHBSRAM2			
	0x30000000 - 0x30003FFF	AHBSRAM1			
	0x2FFF0000 - 0x2FFFFFFF	AXISRAM4			
	0x2FFE0000 - 0x2FFEFFFF	AXISRAM3			
	0x2FFD0000 - 0x2FFDFFFF	AXISRAM2			
	0x2FFB0000 - 0x2FFCFFFF	AXISRAM1			
	0x20020000 - 0x2FFAFFFF	Reserved			
	0x20000000 - 0x2001FFFF	DTCM			

Table 6. Memory map and default device memory area attribute

Region	Boundary address	Description	Type	Attributes	Execute never
Code	0x1FF20000 - 0x1FFFFFFF	Reserved	Normal	Write-through cache attribute	No
	0x1FF00000 - 0x1FF1FFFF	System flash			
	0x90000000 - 0x1FEFFFFF	Reserved			
	0x08FFFFF0 - 0x08FFFFFF	Flash Read Only			
	0x08FFFA00 - 0x08FFFFEF	Reserved			
	0x08FFF800 - 0x08FFF9FF	Flash Read Only			
	0x08FFF400 - 0x08FFF7FF	Reserved			
	0x08FFF000 - 0x08FFF3FF	User OTP			
	0x08200000 - 0x08FFEFFF	Reserved			
	0x08100000 - 0x081FFFFF	User flash memory bank 2			
	0x08000000 - 0x080FFFFF	User flash memory bank 1			
	0x00030000 - 0x07FFFFFF	Reserved			
	0x00000000 - 0x0002FFFF	ITCM			

Table 7. Peripheral boundary addresses(1)

Boundary address	Peripheral	Bus	Registers
0x5C00A000 - 0x5FFFFFFF	Reserved	Debug	Reserved
0x5C009000 - 0x5C009FFF	ETF		Debug registers[TBD]
0x5C008000 - 0x5C008FFF	Replicator		
0x5C007000 - 0x5C007FFF	TPIU		
0x5C006000 - 0x5C006FFF	CSTF		
0x5C005000 - 0x5C005FFF	CTI2		
0x5C004000 - 0x5C004FFF	CTI1		
0x5C003000 - 0x5C003FFF	SWO		
0x5C002000 - 0x5C002FFF	Reserved		
0x5C001000 - 0x5C001FFF	DBGMCU		
0x5C000000 - 0x5C000FFF	SYSROM1 ROM table		
0x58028000 - 0x5BFFFFFF	Reserved	AHB4	Reserved
0x58027000 - 0x58027FFF	RAMECC2		*RAMECC registers*
0x58025000 - 0x58026FFF	Reserved		Reserved
0x58024C00 - 0x58024FFF	CRC		*CRC registers*
0x58024800 - 0x58024BFF	PWR		*PWR registers*
0x58024400 - 0x580247FF	RCC		*RCC registers*
0x58022400 - 0x580243FF	Reserved		Reserved
0x58022000 - 0x580223FF	GPIOI		*GPIO registers*
0x58021C00 - 0x58021FFF	GPIOH		
0x58021800 - 0x58021BFF	GPIOG		
0x58021400 - 0x580217FF	GPIOF		
0x58021000 - 0x580213FF	GPIOE		
0x58020C00 - 0x58020FFF	GPIOD		
0x58020800 - 0x58020BFF	GPIOC		
0x58020400 - 0x580207FF	GPIOB		
0x58020000 - 0x580203FF	GPIOA		
0x58010000 - 0x5801FFFF	Reserved		Reserved

Table 7. Peripheral boundary addresses(1) (continued)

Boundary address	Peripheral	Bus	Registers
0x58008000 - 0x5800FFFF	Reserved	APB4	Reserved
0x58007E00 - 0x58007FFF	COMP8		*COMP registers*
0x58007C00 - 0x58007DFF	COMP7		
0x58007A00 - 0x58007BFF	COMP6		
0x58007800 - 0x580079FF	COMP5		
0x58007600 - 0x580077FF	COMP4		
0x58007400 - 0x580075FF	COMP3		
0x58007200 - 0x580073FF	COMP2		
0x58007000 - 0x580071FF	COMP1		
0x58006C00 - 0x58006FFF	Reserved		Reserved
0x58006800 - 0x58006BFF	DTS		*DTS registers*
0x58004C00 - 0x580067FF	Reserved		Reserved
0x58004800 - 0x58004BFF	IWDG		*IWDG registers*
0x58004400 - 0x580047FF	TAMP		*TAMP registers*
0x58004000 - 0x580043FF	RTC		*RTC registers*
0x58003C00 - 0x58003FFF	VREFBUF		*VREFBUF registers*
0x58003400 - 0x58003BFF	Reserved		Reserved
0x58003000 - 0x580033FF	LPTIM5		*LPTIM registers*
0x58002C00 - 0x58002FFF	LPTIM4		
0x58002800 - 0x58002BFF	LPTIM3		
0x58002400 - 0x580027FF	LPTIM2		
0x58001800 - 0x580023FF	Reserved		Reserved
0x58001400 - 0x580017FF	SPI6		*SPI/I2S registers*
0x58001000 - 0x580013FF	Reserved		Reserved
0x58000C00 - 0x58000FFF	LPUART1		*LPUART registers*
0x58000800 - 0x58000BFF	Reserved		Reserved
0x58000400 - 0x580007FF	SBS		*SBS registers*
0x58000000 - 0x580003FF	EXTI		*EXTI registers*

Table 7. Peripheral boundary addresses(1) (continued)

Boundary address	Peripheral	Bus	Registers
0x52100000 - 0x57FFFFF	Reserved	AHB5	Reserved
0x520E0000 - 0x520FFFFF	NPU		NPU registers[TBD]
0x5200A000 - 0x520DFFFF	Reserved		Reserved
0x52009000 - 0x52009FFF	RAMECC1		*RAMECC registers*
0x52008400 - 0x52008FFF	Reserved		Reserved
0x52008000 - 0x520083FF	DLYBSD1		SDMMC delay buffer registers[TBD]
0x52007000 - 0x52007FFF	SDMMC1		*SDMMC registers*
0x52006400 - 0x52006FFF	Reserved		Reserved
0x52006000 - 0x520063FF	DLYBXS1		XSPI delay buffer registers[TBD]
0x52005000 - 0x52005FFF	XSPI1		*XSPI registers*
0x52004000 - 0x52004FFF	FMC		*FMC register map*
0x52003000 - 0x52003FFF	Reserved		Reserved
0x52002000 - 0x52002FFF	FLASH		*FLASH registers*
0x52001000 - 0x52001FFF	Reserved		Reserved
0x52000000 - 0x52000FFF	HPDMA1		*HPDMA registers*
0x50000000 - 0x51FFFFFF	Reserved		Reserved
0x48004000 - 0x4FFFFFFF	Reserved	AHB3	Reserved
0x48028000 - 0x4803FFFF	Reserved		Reserved
0x48027C00 - 0x48027FFF	DAC4		*DAC registers*
0x48027800 - 0x48027BFF	DAC3		
0x48027400 - 0x480277FF	DAC2		
0x48027000 - 0x480273FF	DAC1		
0x48026800 - 0x48026FFF	Reserved		Reserved
0x48026400 - 0x480267FF	ADC34		*ADC registers (for each ADC)* *ADC common registers*
0x48026000 - 0x480263FF	ADC12		
0x48020800 - 0x48025FFF	Reserved		Reserved
0x48020400 - 0x480207FF	HASH		*HASH registers*
0x48020000 - 0x480203FF	RNG		*RNG registers*
0x48004C00 - 0x4801FFFF	Reserved	AHB2	Reserved
0x48004800 - 0x48004BFF	FMAC1		*FMAC registers*
0x48004400 - 0x480047FF	CORDIC		*CORDIC registers*
0x48000000 - 0x480043FF	Reserved		Reserved

Table 7. Peripheral boundary addresses(1) (continued)

Boundary address	Peripheral	Bus	Registers
0x42006000 - 0x47FFFFFF	Reserved	APB2	Reserved
0x42007000 - 0x42007FFF	HRTIM2		*HRTIM registers*
0x42006000 - 0x42006FFF	HRTIM1		
0x42005400 - 0x42005FFF	Reserved		Reserved
0x42005000 - 0x420053FF	SPI5		*SPI/I2S registers*
0x42004C00 - 0x42004FFF	TIM9		*TIM9/TIM12 registers*
0x42004800 - 0x42004BFF	TIM17		*TIM16/TIM17 registers*
0x42004400 - 0x420047FF	TIM16		
0x42004000 - 0x420043FF	TIM15		
0x42003800 - 0x42003FFF	Reserved		Reserved
0x42003400 - 0x420037FF	SPI4		*SPI/I2S registers*
0x42003000 - 0x420033FF	SPI1		
0x42002000 - 0x42002FFF	Reserved		Reserved
0x42001C00 - 0x42001FFF	USART10		*USART registers*
0x42001800 - 0x42001BFF	UART9		
0x42001400 - 0x420017FF	USART6		
0x42001000 - 0x420013FF	USART1		
0x42000C00 - 0x42000FFF	Reserved		Reserved
0x42000800 - 0x42000BFF	TIM20 (PWM)		*TIM1/TIM8/TIM20 registers*
0x42000400 - 0x420007FF	TIM8 (PWM)		
0x42000000 - 0x420003FF	TIM1 (PWM)		
0x400C0000 - 0x41FFFFFF	Reserved	AHB1	Reserved
0x40080000 - 0x400BFFFF	OTG_FS		*OTG_FS registers*
0x40030000 - 0x4007FFFF	Reserved		Reserved
0x4002F000 - 0x4002FFFF	ADF1		*ADF registers*
0x40022000 - 0x4002EFFF	Reserved		Reserved
0x40021000 - 0x40021FFF	GPDMA1		*GPDMA registers*
0x40020C00 - 0x40020FFF	HSP1		*HSP registers*
0x40020000 - 0x40020BFF	Reserved		Reserved

Table 7. Peripheral boundary addresses(1) (continued)

Boundary address	Peripheral	Bus	Registers
0x4000F000 - 0x4001FFFF	Reserved	APB1	Reserved
0x4000EC00 - 0x4000EFFF	UCPD1		*UCPD registers*
0x4000B800 - 0x4000EBFF	Reserved		Reserved
0x4000AC00 - 0x4000B7FF	FDCANRAM		FDCAN SRAM
0x4000A800 - 0x4000ABFF	FDCAN3		*FDCAN registers*
0x4000A400 - 0x4000A7FF	FDCAN2		
0x4000A000 - 0x4000A3FF	FDCAN1		
0x40008800 - 0x40009FFF	Reserved		Reserved
0x40008400 - 0x400087FF	CRS		*CRS registers*
0x40008000 - 0x400083FF	Reserved		Reserved
0x40007C00 - 0x40007FFF	UART8		*USART registers*
0x40007800 - 0x40007BFF	UART7		
0x40006400 - 0x400077FF	Reserved		Reserved
0x40006000 - 0x400063FF	I3C1		*I3C registers*
0x40005C00 - 0x40005FFF	I2C3		*I2C registers*
0x40005800 - 0x40005BFF	I2C2		
0x40005400 - 0x400057FF	I2C1		
0x40005000 - 0x400053FF	UART5		*USART registers*
0x40004C00 - 0x40004FFF	UART4		
0x40004800 - 0x40004BFF	USART3		
0x40004400 - 0x400047FF	USART2		
0x40004000 - 0x400043FF	Reserved		Reserved
0x40003C00 - 0x40003FFF	SPI3		*SPI/I2S registers*
0x40003800 - 0x40003BFF	SPI2		
0x40003400 - 0x400037FF	TIM11		*TIM10/TIM11/TIM13/TIM14 registers*
0x40003000 - 0x400033FF	TIM10		
0x40003000 - 0x400037FF	Reserved		Reserved
0x40002C00 - 0x40002FFF	WWDG		*WWDG registers*
0x40002800 - 0x40002BFF	Reserved		Reserved
0x40002400 - 0x400027FF	LPTIM1		*LPTIM registers*
0x40002000 - 0x400023FF	TIM14		*TIM10/TIM11/TIM13/TIM14 registers*
0x40001C00 - 0x40001FFF	TIM13		
0x40001800 - 0x40001BFF	TIM12		
0x40001400 - 0x400017FF	TIM7 (Basic)		*TIM6/TIM7 registers*

Table 7. Peripheral boundary addresses(1) (continued)

Boundary address	Peripheral	Bus	Registers
0x40001000 - 0x400013FF	TIM6 (Basic)	APB1	*TIM6/TIM7 registers*
0x40000C00 - 0x40000FFF	TIM5 (32b)		*TIM2/TIM3/TIM4/TIM5 registers*
0x40000800 - 0x40000BFF	TIM4 (32b)		
0x40000400 - 0x400007FF	TIM3 (32b)		
0x40000000 - 0x400003FF	TIM2 (32b)		

1. Accessing a reserved area results in a bus error. Accessing undefined memory space in a peripheral returns zeros.

2.4 Embedded SRAM

The STM32H7Pxxx devices include 736 Kbytes of SRAM.

- 192 Kbytes of tightly coupled instruction memory (ITCM) with ECC.
- 128 Kbytes of tightly coupled data memory (DTCM) with ECC.
- 320 Kbytes of AXISRAM with ECC, accessible via the AXI bus directly.
- 96 Kbytes of AHBSRAM without ECC, accessible via the AXI and AHB interconnects.

The TCM SRAMs are dedicated to the Cortex®-M7:

- DTCM is mapped at the address 0x2000 0000 and accessible by Cortex®-M7 via a dedicated dual 32-bit interface. This allows simultaneous 32-bit accesses, or single 64-bit accesses. It can also be accessed at lower priority by the HPDMA, GPDMA, and NPU through the AHBS port of the Cortex®-M7 CPU, thereby eliminating the overhead of transferring data by software from system memory or peripherals.
- ITCM is mapped at the address 0x0000 0000 and accessible by Cortex®-M7 via a single 64-bit dedicated interface. It is also accessible at lower priority by HPDMA (and GPDMA) through the AHBS port of the Cortex®-M7 CPU. The ITCM can be used to host code for time-critical routines (such as interrupt handlers), or that requires deterministic execution.

The AXISRAM comprises four blocks:

- AXISRAM1 - 128 Kbytes
- AXISRAM2/3/4 - 3 x 64 Kbytes

AXISRAM supports 8-bit, 16-bit, 32-bit, and 64-bit accesses. It can be used for code, large data buffers, general variables, etc. It is optimized for high data throughput when accessed by the Cortex®-M7 CPU, the HPDMA, or the NPU.

AXISRAM1 can be automatically erased in the event of a tamper. It is also erased systematically by hardware after a power-on reset. This feature makes AXISRAM1 suitable for temporary storage of sensitive information in clear (unencrypted) format.

The AHBSRAM comprises three blocks:

- AHBSRAM1/2 - 2 × 16 Kbytes
- AHBSRAM3 - 64 Kbytes

AHBSRAM supports 8-bit, 16-bit, and 32-bit accesses. It is intended for local buffering of data coming from or destined for peripherals such as the ADCs, DACs, serial communications interfaces. Being able to store data locally avoids the need to interrupt the CPU frequently to perform transfers, and optimizes the traffic on the AXI by allowing

occasional large bursts rather than frequent short bursts. Nevertheless, it can also be used by the CPU and the NPU for additional storage, albeit with lower throughput.

AHBSRAM3 is dedicated to the HSP when it is enabled. Part of AHBSRAM3 is aliased as HSPBRAMAB, which is used for data exchange between the HSP and the system (see chapter [HSP]). When the HSP is disabled, the AHBSRAM3 can be used as system SRAM.

The backup SRAM (BKPSRAM) can be retained in all low-power modes, and when V_{DD} is off in V_{BAT} mode with a battery connected to the VBAT pin. Refer to *Section 6.4.3: Backup domain* for more details. The backup SRAM is protected by the tamper detection circuit, and are erased by hardware in case of tamper detection. They are also erased by hardware in case of a backup domain reset. Refer to (link to *Section 43: Tamper and backup registers (TAMP)*) for more details.

Table 8 summarizes the internal SRAM resources:

Table 8. SRAM summary

SRAM	STM32H7Pxxx
ITCM	192 Kbytes with ECC. Tightly coupled to CPU.
DTCM	128 Kbytes with ECC. Tightly coupled to CPU.
AXISRAM1	128 Kbytes with ECC. Automatic erase at power on and in the event of a tamper.
AXISRAM2	64 Kbytes with ECC.
AXISRAM3	64 Kbytes with ECC.
AXISRAM4	64 KBytes with ECC.
AHBSRAM1	16 Kbytes, no ECC.
AHBSRAM2	16 Kbytes, no ECC.
AHBSRAM3	64 Kbytes, no ECC. Dedicated to HSP when enabled.
BKPSRAM	4 Kbytes with ECC. Battery backed-up. Automatic erase in the event of a tamper, a backup domain reset or a life-cycle regression.

ECC error detection is handled by the RAMECC units. Refer to *Section 4: RAM ECC monitoring (RAMECC)*.

2.5 Flash memory overview

The flash memory interface manages accesses to the flash nonvolatile memory. It implements the erase and program flash memory operations and the read and write protection mechanisms.

The flash memory is organized as follows:

- Dual bank main memory block divided into sectors.
  - Supports reading from one bank while reprogramming the other
- An information block:
  - System memory from which the device boots in system memory boot mode
  - Option bytes to configure read and write protection, BOR level, watchdog software/hardware and reset when the device is in Standby or Stop mode.

Refer to *Section 5: Embedded flash memory (FLASH)* for further details.

2.6 Boot configuration

In the STM32H7Pxxx, two different boot areas can be selected through the BOOT0 pin as shown in the *Table 9*.

Table 9. Boot modes

Boot mode selection			Boot area
PRODUCT_STATE	**NVSTATE**	**BOOT0**	
Open	OPEN	0	Boot from the user flash memory at 0x0800 0000
		1	Boot from the bootloader
Provisioning	CLOSE	-	Boot from the RSS in system flash memory Then jump into the bootloader
Closed/Locked			Boot from the RSS in system flash memory Then jump into the user flash memory at 0x0800 0000(1)

1. MPU is enabled when executing OEM firmware in user flash memory; depending on needed resources, OEM firmware should update MPU configuration or disable MPU.

The values on the BOOT0 pin are latched after the power-on reset is released. The user must set the BOOT0 pin level during power-on reset.

The BOOT0 pin is resampled when the device exits Standby mode. Consequently, the values must be kept in the required Boot mode configuration when the device is in Standby mode.

After startup delay, the selection of the boot area is done before releasing the processor reset.

Embedded bootloader

The embedded bootloader code is located in system memory. It is programmed by ST during production. It is used to reprogram the flash memory using one of the following serial interfaces:

- DFU: USB OTG_FS in device mode
- USART1
- USART2
- USART3
- UART4
- FDCAN2
- I2C1
- I2C2
- I2C3
- I3C1
- SPI1
- SPI2
- SPI3

For additional information, refer to *Section 10: System configuration, boot and security (SBS)*.

3 System security

The microcontrollers of the STM32H7Pxxx series are designed with a comprehensive set of security features, which simplify the evaluation of IoT devices against security standards. These features also reduce the cost and complexity of software development for OEMs and third-party developers by facilitating reuse, improving interoperability, and minimizing API fragmentation.

This section describe the security features available on STM32H7Pxxx devices.

3.1 Key security features

- Device identification with device 96-bit unique ID and JTAG 32-bit die-specific ID.
- Boot levels isolation enabled by the HDPL (hide protection level) monotonic counter.
- Resource isolation using the privilege mode of Cortex-M7, extended to HPDMA, GPDMA, RTC, and TAMP peripherals.
- Additional hardware features for secure boot.
  - Unique boot entry in embedded flash memory.
  - Embedded flash memory write protection.
  - Nonvolatile user option bytes dedicated to bank swapping, write protection, and HDP areas control.
- Secure storage, featuring:
  - Boot data storage located in HDP-isolated areas of the embedded flash (option byte keys).
  - Battery-powered volatile secure storage, automatically erased in case of tamper.
- Cryptographic true random number generation, NIST SP800-90B compliant.
- General purpose cryptographic acceleration.
  - HASH processor, supporting SHA-1 checksums and SHA-2 secure hash (SHA-256, SHA-384, SHA-512).
- Device life cycle management, which includes debug and regression control.
- Security services in system flash memory
- Active tamper and protection against temperature, voltage, and frequency attacks.
  - Flexible active tamper I/O management: from three meshes (each input associated with its own exclusive output) to six meshes (single output shared for up to five tamper inputs). Some tamper I/Os remain functional in V_{BAT} mode.

3.2 Identification

3.2.1 Unique ID

The devices store a 96-bit ID that is unique to each device (see *Section 53.1: Unique device ID register (96 bits)*). The application can use this unique identity key to identify the product in the cloud network or make it difficult for counterfeit devices or clones to inject untrusted data into the network.

3.2.2 Discover command

The debug protocol included in the device's immutable code contains a "DISCOVER" command that provides basic information about the target: manufacturer, product name, device identifier, supported protocol version, and available cryptographic scheme for authentication (password only).

3.3 Secure boot

Secure boot is immutable code that always executes after a system reset when the device is not in the open state (see *Section 3.8.1*). As the root of trust, this code verifies the device's static protections and activates available runtime protections, reducing the risk of invalid or malicious code running on the platform. Additionally, secure boot checks the integrity and authenticity of the next-level firmware before executing it.

This section details the features specifically designed for secure boot.

3.3.1 Unique boot entry

As defined in *Section 10.3.5: SBS boot control*, the device has two booting methods after each reset:

- For closed devices (NVSTATE = NVCLOSE): The device boots on a small ST immutable code in the system flash, which jumps to a fixed address in the user flash (address 0x0800 0000).
- For open devices (NVSTATE = NVOPEN): device boots from a fixed address in the user flash (BOOT pin = 1) or from the bootloader in the system flash (BOOT pin = 0).

When the device's nonvolatile state (NVSTATE) equals NVCLOSE, it is recommended that the code located at address 0x0800 0000 is write protected (see *Section 3.3.2*).

When the swapping function is activated (SWAP_BANK option in the FLASH_OBW1SR register), a different physical bank is allocated to the user space address 0x0800 0000. Swapping can be disabled with the nonvolatile BOOT_LOCK option in FLASH_ROTSR register. The FLASH_ROTSR register can only be configured using RSSLIB SetSecOB function. Refer to *Section 3.9.2* for details.

3.3.2 Embedded flash memory write protection

The embedded flash memory in the device includes write protection (WRP) to prevent unwanted modifications to selected embedded nonvolatile code or data.

Typical usage of the write protection is to protect the integrity of the boot code stored at the unique boot entry in the user flash. This protection can only be deactivated by the OEM boot code itself, running in HDP level 1. Refer to *Section 5.5.2: Write protection* for details.

As a write-protected area cannot be erased or programmed, a full bank erase cannot be performed if any group of four consecutive pages is write-protected, unless a transition to Open is triggered (see **Regression** in *Section 3.8.1*).

Note: The HDP area in user flash is write- and erase-protected, while system flash cannot be erased by the user application.

3.3.3 Boot levels isolation with HDP

Section 3.4.1 explains how HDPL (hide protection level) hardware isolates boot levels.

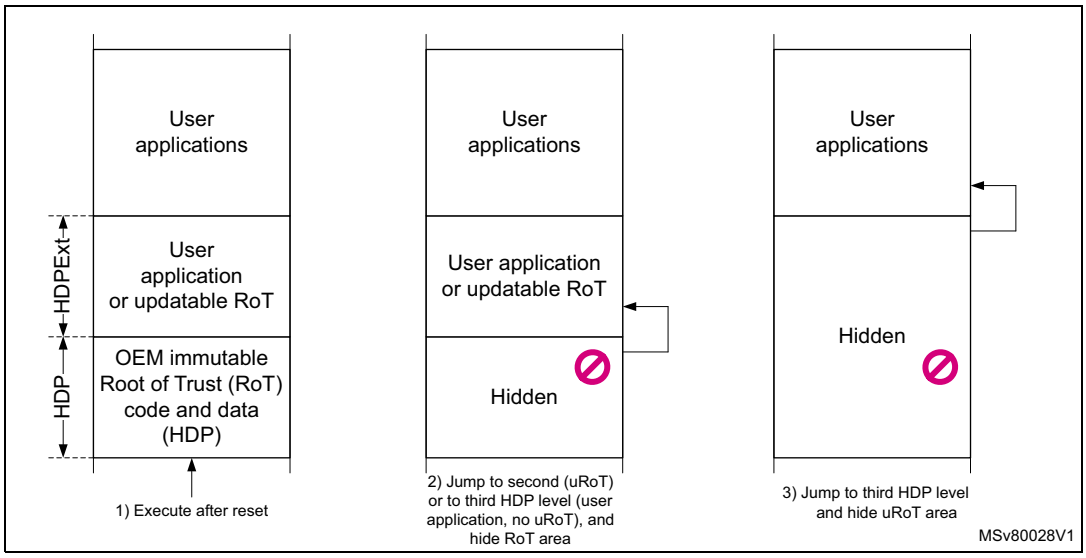
3.4 Isolation

The microcontrollers in the STM32H7 series support isolation. Isolating a trustworthy service from less trusted or untrusted services is essential to protect its integrity. This section describes the features specifically designed for isolation.

3.4.1 Temporal isolation using hide protection levels (HDPL)

The hardware and software resources used during each boot level can be isolated. This process relies on a Hide Protection (HDP) monotonic counter that only increments the levels. When the counter increments, the resources from previous levels, including both code and data, become hidden. This feature, summarized in *Figure 4*, is described in *Section 5.5.4: Hide protection (HDP)* and *Section 10.3.4: SBS hide protection management* of this document.

Figure 4. User flash memory HDP area principle



For each bank, activation of the HDP area in user flash memory is linked to HDPL1. Once HDPL is equal to or greater than 2, data read, write, and instruction fetch accesses to the HDP area defined by the HDP_AREA_START and HDP_AREA_END option bytes are denied until the next device reset.

The end of each HDP area can be dynamically extended by the boot application by setting the FLASH_xHDPEXTR register (x = A or B). This extension prevents access to protected sectors used by updatable boot code when runtime code executes in HDPL3.

Note: A bank erase operation aborts if it contains a write-protected area (WRP or HDP area).

3.4.2 Isolation using privileged mode

The hardware and software resources of STM32H7Pxxx devices can be partitioned to restrict access to software running in privileged mode.

This hardware isolation technology protects critical code, data and specific hardware resources against intentional or unintentional tampering.

Memory and peripheral privileged allocation using MPU

The Cortex-M7 includes a memory protection unit (MPU) that restricts read and write accesses to memory regions, including regions mapped to peripherals, based on one or more of the following parameters:

- Cortex-M7 operating mode (privileged or unprivileged).
- Data or instruction fetch.

The memory map and the MPU configuration divide memory into up to sixteen regions, each aligned to a multiple of 32 bytes. Each memory region can be programmed to generate faults when accessed inappropriately by unprivileged software.

Managing accesses in privileged-aware peripherals

The following peripherals implement privileged-only access mode. Peripherals not listed can be read and written by both privileged and unprivileged accesses.

- *Direct memory access controllers (HPDMA and GPDMA)*

  When a DMA channel x is defined as privileged (PRIVx = 1 in xDMA_PRIVCFGR), specific rules (see *Table 10*) apply when accessing privileged or unprivileged sources or destinations.

Table 10. DMA channel use (privilege)

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

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

- *Real time clock (RTC)*

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

  When the PRIV bit is set in privileged-only RTC_PRIVCFGR register:

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

  – Reading the RTC_PRIVCFGR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER, and RTC_CALR registers is always possible in privileged and unprivileged modes. All other RTC registers can be read only in privileged mode.

  When the PRIV bit is cleared in privileged-only RTC_PRIVCFGR register, it is still possible to restrict access to privileged mode for certain RTC registers by setting

dedicated control bits: INITPRIV, CALPRIV, TSPRIV, WUTPRIV, ALRAPRV, or ALRBPRIV.

See *Section 42: Real-time clock (RTC)* for details.

- *Tamper and backup registers (TAMP)*

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

  When the PRIV bit is set in the privileged-only TAMP_PRIVCFGR register:

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

  – Reading the TAMP_PRIVCFGR register is always possible in both privilege and unprivileged modes. All other TAMP registers can be read only in privilege mode, except for the backup registers and the monotonic counters, which have their own protection settings.

  The application can also:

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

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

    – Make protection zone 1 read-privileged and write-privileged.

    – Make protection zone 2 read-privileged or unprivileged and write-privileged.

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

  See *Section 42: Real-time clock (RTC)* for details.

3.5 Secure storage

A critical feature of security systems is how long-term keys are stored, protected, and provisioned. These keys are typically used for loading a boot image or managing critical user secrets, such as the regression password.

3.5.1 Boot data storage

As described in *Section 5.4.4: Option byte key management* the device manages 32 key slots of 16-bytes per HDP level (up to level 2, inclusive). A customer application running in HDPL1 can provision keys in both HDPL1 and HDPL2 by using NEXTKL[1:0] field in the FLASH_OBKCR register. An application running in HDPL2 can only provision or update its own keys. HDP protection is detailed in *Section 3.4.1*.

3.5.2 Battery-powered volatile secure storage

The device includes 32 backup registers located in the backup domain, which remain powered by V_{BAT} when the V_{DD} power is switched off. These registers are protected against tampering, as described in *Section 3.7*.

3.6 Cryptographic engines

3.6.1 Cryptographic random number generation

The RNG peripheral in the device is a NIST-compliant true random number generator that provides full-entropy outputs to the application. For more information, refer to *Section 31: True random number generator (RNG)*.

3.6.2 Cryptographic operations

Table 11 lists the accelerated cryptographic operations available in the device. For more information, refer to *Section 32: Hash processor (HASH)*.

Table 11. Device cryptographic operations

Operations	Algorithm	Specification	Length of keys (in bits)	Modes
Secure hash	SHA-2	FIPS PUB 180-4	N/A	SHA-256, SHA-384, SHA-512
Keyed-hashing for message authentication	HMAC	FIPS PUB 198-1 IETF RFC 2104	Short, long (>64 bytes)	-

3.7 Tamper detection and response

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

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

See *Section 43: Tamper and backup registers (TAMP)* for more details.

3.7.1 Tamper detection sources

The devices support seven active input/output pins, enabling three independent active-tamper meshes or up to six meshes if the same output pin is shared by six meshes.

The active pins are clocked by the LSE and function in various system operating modes: Run, Sleep, Stop, or Standby. Refer to *Table 496: TAMP pins functionality over modes* for a list of tamper pins and their availability, including in V_{BAT} mode.

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

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

The internal tamper sources are listed in *Table 12*.

Table 12. Internal tampers in TAMP

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

3.7.2 Response to tampers

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

- Generate an interrupt: This interrupt is capable of waking up the device from Stop and Standby modes (see the TAMPxMSK bits in the TAMP_CR2 register).
- Generate a hardware trigger: This trigger is for the low-power timers.
- Erase device secrets: This occurs if the corresponding TAMPxNOER bit is cleared in TAMP_CR2 (for external tamper pins) or TAMP_CR3 (for internal tampers). The erasable secrets include:
  - Symmetric keys stored in backup registers (x32), or in HASH
  - Secrets stored in AXI SRAM1
  - Backup SRAM (depending on configuration bit), erased when V_{DD} is present

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

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

Note: Device secret erase is also triggered by setting the BKERASE bit in TAMP_CR2 or by performing a PRODUCT_STATE regression as defined in Section 3.8.

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

3.7.3 Software filtering mechanism

Each tamper source can be configured to avoid launching an immediate erase by setting the corresponding TAMPxNOER bit in TAMP_CR2 (for external tamper pins) or TAMP_CR3 (for internal tampers). In this situation, when the tamper flag is raised, the following mechanisms optionally activate until all tamper flags are cleared.

- Backup registers, backup SRAM, AXI SRAM1: Read operation returns 0 and write operations are ignored.
- HASH peripherals: Reset by RCC.

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

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

Note: If the tamper software fails to react to the tamper flag, the tamper input associated with the IWDG reset automatically triggers the erase of secrets.

3.7.4 Tamper detection and low-power modes

The effects of low-power modes on tamper detection are summarized in *Table 13*.

Table 13. Effect of low-power modes on TAMP

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

3.8 Product life cycle

A typical IoT device life cycle is summarized in *Figure 5*. For each step, the devices provide secure life cycle management mechanisms embedded in the hardware. *Table 14* details the various phases and their associated transitions.

Figure 5. Device life cycle security

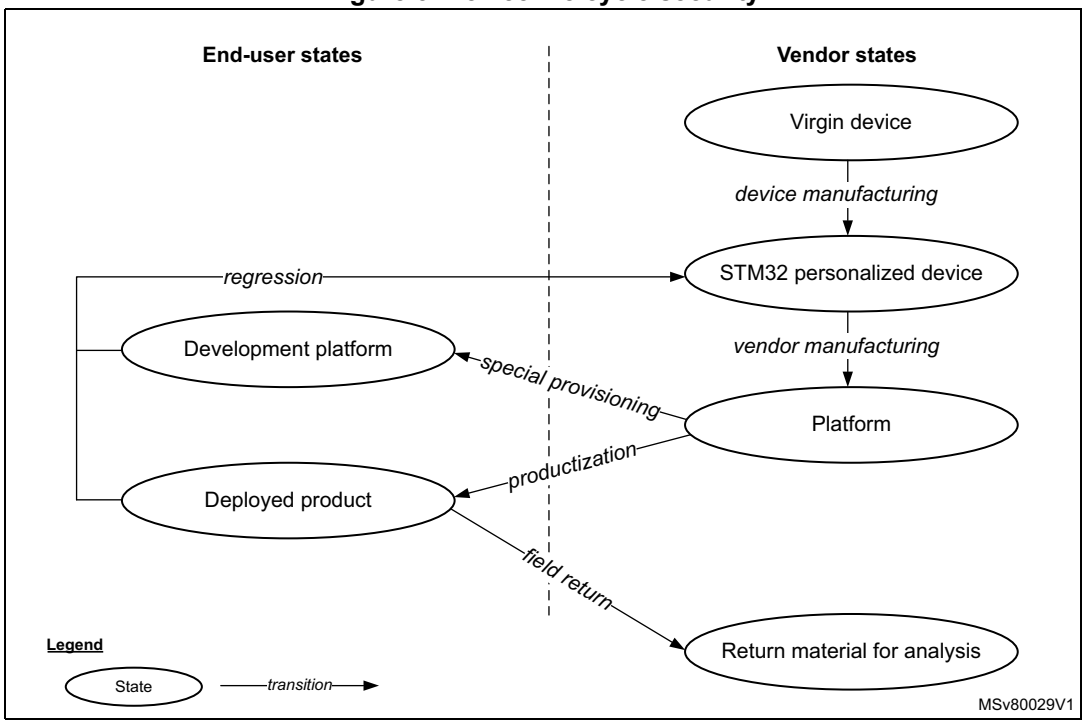


Table 14. Main product life cycle transitions

Transitions	Description
Device manufacturing	ST creates new STM32 devices, always checking for manufacturing defects. During this process, the device is provisioned with ROM firmware, and a public ID.
Vendor manufacturing	One or more vendors are responsible for the platform assembly, initialization, and provisioning before delivery to the end user. This end user can use the final product (referred to as the “productization” transition) or use the platform for software development (referred to as “special provisioning” transition).
Productization	The end user gets a product ready for use. All security functions of the platform are enabled, debugging and testing features are restricted or disabled, and a unique boot entry to immutable code is enforced.
Special provisioning	The platform vendor prepares an individual platform for development, ensuring it is not connected to a production cloud network.
Field return	The end user returns the device as material for analysis and ships it back to the manufacturer. During this process, all data, including user data, is destroyed. Consequently, the devices loses the ability to operate securely, such as connecting to a managed IoT network. This transition is not possible when the PRODUCT_STATE is Locked.

The features described in this section help secure the device life cycle.

3.8.1 Life cycle management

The devices implement the life cycle described in *Section 5.5.3: Life cycle management*. For example, secure boot is enforced when NVSTATE equals NVCLOSE.

Note: STMicroelectronics does not analyze defective parts in the locked state (field return transition in Table 14).

Debug management

As explained in *Section 10.3.6: SBS debug management*, device debugging may or may not be allowed, depending on the device life cycle:

- For open devices (NVSTATE = NVOPEN), debugging is always allowed.
- For closed devices (NVSTATE = NVCLOSE), debugging is disabled at boot with a policy defined by the option byte DBG_AUTH[7:0] in the FLASH_ROTSR register. Possible options are:
  - 0xB4: Locked device (no debugging, no regression)
  - 0x8A: Closed device (no debugging).

The FLASH_ROTSR register can only be configured using RSSLIB SetSecOB function. Refer to *Section 3.9.2* for details.

Regression

When the device is in the “closed” or “provisioning” product state, regression to the “open” state is possible using the 128-bit regression password provisioned during the manufacturing phase of the final product. Upon receiving a valid regression request, the device erases all the OEM secrets, except the regression password, before changing the product state.
Refer to **Flash opening** in *Section 5.5.3: Life cycle management* for details on this opening sequence. Details on the tools available to perform regressions is available in TBD ANxxx.

3.9 System memory

System memory stores root secure services (RSS) firmware programmed by STMicroelectronics during production. The RSS provides runtime services to user firmware when permitted.

3.9.1 RSS user functions

The runtime services are provided by the RSS library. The functions of the RSS library, listed in *Table 16*, are accessible to users through the CMSIS device header file included in the STM32CubeH7P firmware package.

Table 15. Macros for RSS services

C defined macro	Location in system memory
RSSLIB_PFUNC	0x1FF1FD4C

Before calling RSSLIB functions, if MPU is enabled, the user firmware must:

- Define an MPU region to allow read access to the location starting from RSSLIB_SYS_FLASH_PFUNC_START (0x1FF1FD4C) up to RSSLIB_SYS_FLASH_PFUNC_END (0x1FF1FD78). These addresses are provided in the CMSIS device header file.
- Define an MPU region to allow execution of RSSLIB firmware, starting from 0x1FF0E000 up to 0x1FF0FFFF.

3.9.2 RSSLIB

The user firmware calls RSSLIB functions using the RSSLIB_PFUNC C-defined macro.

Table 16. RSS lib interface functions

Library	Callable function
RSSLIB_PFUNC	SetSecOB(1)
	GetRssStatus(1)
	SetProductState(1)
	DataProvisioning(1)
	JumpHDPLvl2
	JumpHDPLvl3
	GetProductState(1)

1. Functions used for OEM personalization are performed using provisioning tools.

SetSecOB

Prototype:

```
uint32_t SetSecOB(uint32_t ObAddr, uint32_t ObMask, uint32_t
ObValue, uint32_t ObPos)
```

User code function call example:

```
RSSLIB_PFUNC->SetSecOB((uint32_t) ObAddr, (uint32_t) ObMask,
(uint32_t) ObValue, (uint32_t) ObPos);
```

Arguments:

- `ObAddr`
  - Input parameter: address of the secure OB to configure. It must be one of the following (x = A or B):

    &(FLASH->xWRPSRP)

    &(FLASH->xHDPSRP)

    &(FLASH->ROTSRP)
- `ObMask`
  - Input parameter, mask of the OB part to be updated (for example, 0x00000FFF for updating HDP_AREA_START OB).
- `ObValue`
  - Input parameter: OB value to be applied.
- `ObPos`
  - Input parameter: Apply the OB bit shift to the ObValue. Ensure that (ObValue<<ObPos) aligns on ObMask.

The user calls the SetSecOB function to enable RSS to configure the secure option bytes after a reset.

Note: This function is available only in the provisioning state of the product.

Returned values:

- On a successful call, the function does not return and does not push LR onto the stack.
- In case of failure (bad product state), SetSecOB returns 0xF5F5F5F5.

To retrieve the OB change status (after RSS execution), call the GetRssStatus function or read back the OB. For example, read the FLASH_AWRPSR register.

GetRssStatus

Prototype:

```
uint32_t GetRssStatus(void)
```

User code function call example:

```
RSSLIB_PFUNC->GetRssStatus();
```

Arguments: None

The user calls `GetRssStatus` to retreive the RSS execution status of the previous RSSLIB function call that triggerred a reset for execution by RSS (`SetSecOB` or `DataProvisioning`).

Returned values:

`0xEAEAEAEA`	Success
`0xF5F5F5F5`	Error

0xF5F5E0E0	Bad input address range
0xF5F50E0E	Bad size
0xF5F5E00E	Bad encryption value in DoEncryption
0xF5F50808	Programming error
0xF5F50008	Bad value in Destination

SetProductState

Prototype:

```
uint32_t SetProductState(uint32_t ProductState)
```

User code function call example:

```
RSSLIB_PFUNC->SetProductState((uint32_t) ProductState);
```

Arguments:

- ProductState
  - Input parameter: STM32 product state to reach. The parameter must be one of the following:

    0x17: Provisioning product state. This state is allowed when the current product state is Open.

    0x72: Closed product state. This state is allowed when current product state is provisioning, and data provisioning has been made.
  - 0x5C: Locked product state. This state is allowed when the current product state is provisioning.

The user calls the SetProductState function to change the STM32 product state. The configuration is completed by the RSS after a reset.

Returned values:

- On successful call, the function does not return and does not push LR onto the stack.
- In the case of failure (invalid product state), the GetProductState function returns 0xF5F5F5F5.

To retrieve the new product state after RSS execution, call the GetProductState function.

DataProvisioning

Prototype:

```
uint32_t DataProvisioning(RSSLIB_DataProvisioningConf_t *pConfig)
```

User code function call example:

```
RSSLIB_PFUNC->DataProvisioning((RSSLIB_DataProvisioningConf_t*)
pConfig));
```

Arguments:

- `pConfig`: input parameter RSSLIB_DataProvisioningConf_t , described below:

```
typedef struct
{
  uint32_t *pSource;
  uint32_t Destination;
  uint32_t Size;
  uint32_t DoEncryption;
  uint32_t Crc;
} RSSLIB_DataProvisioningConf_t;
```

With the structure elements:

`pSource`	Provides the address of the data to be provisioned. The address must be within the AXISRAM3 address range.
`Destination`	Provides the destination information, specifying where to provision the data. 0xDADAHHII: – II specifies the OBKey starting index. – HH specifies the OBKey HDPL. – DADA specifies the debug authentication configuration, which is only relevant when the OBKey is writable only in HDPL0.
`Size`	Provides the size of the data to be provisioned (the number of bytes). The size must be a multiple of 16.
`DoEncryption`	The data can only be 0xCACA0AA0U (OBKeys data in the clear)
`Crc`	The CRC is computed over the full source data buffer, **`pConfig→Destination`** value, **`pConfig→Size`** value and finally **`pConfig->DoEncryption`** value. The CRC computation uses CRC-32 (Ethernet) with the following parameters: – CRC polynomial: 0x04C11DB7U – Initial value: 0xFFFFFFFFU

`DataProvisioning` receives a data buffer as input and programs it as OBKeys. A CRC ensures data and parameter integrity to prevent tampering issues.

Returned values:

`0xEAEAEAEAU` Success

`0xF5F5F5F5U` Error: Bad product state.
Data provisioning only available in open and provisioning product states.

`0xF5F5E0E0U` Error: Bad input address range.
`pConfig` or **`pSource`** are not in AXISRAM3.

`0xF5F50E0EU` Error: Bad provisioning size:
– **Size** is not a multiple of 16.
– OBKey starting index + **Size** does not fit within an OBKey section.

`0xF5F58080U` Error: computed CRC is not the expected one provided within **`pConfig→CRC.`**

`0xF5F5E00EU` Error: wrong **`DoEncryption`** parameter value.

`0xF5F50808U` Error: OBKeys programming error.

`0xF5F50008U` Error: Bad destination information:
– OBKey index is not in the range 0 to 31.
– OBKey HDPL is not in the range 0 to 2 (representing HDPL0, HDPL1 and HDPL2).
– OBKey HDPL0 DBG_AUTH (debug authentication) differs from the Closed or Locked states.

`0xF5F58008` Error: Bad HDPL in open product state. In open product state, Data provisioning service is only available in HDPL1.

JumpHDPLvl2

Prototype:

```
uint32_t JumpHDPLvl2(uint32_t VectorTableAddr, uint32_t MPUIndex)
```

User code function call example:

```
RSSLIB_PFUNC->JumpHDPLvl2((uint32_t)NextVectorTableAddr, 1U );
```

Arguments:

- `VectorTableAddr`
  - Input parameter, address of the next vector table to apply
  - The vector table format folllows the one used by the Cortex-M7 core
- `MPUIndex`
  - Input parameter, MPU region index (index 0 to 15 allowed). The caller function must define, but keep disabled, the corresponding MPU region before calling JumpHDPLvl2. The function enables the MPU region before jumping to the reset handler of the vector table. The vector table reset handler function belongs to the MPU region.

The user calls `JumpHDPLvl2` to close the user flash HDPL1 area by incrementing HDPL to 2, then jumps to the reset handler embedded within the vector table, whose address is passed as input parameter.

After closing HDPL1, `JumpHDPLvl2` enables the MPU region provided as an input parameter. Once the MPU is enabled, the function sets the new vector table, sets the SP to the address provided in this new vector table, and jumps to the reset handler function supported by it.

Returned values:

- On successful execution, the function does not return and does not push LR onto the stack.
- In case of failure (invalid input parameter value), JumpHDPLvl2 returns 0xF5F5F5F5.

JumpHDPLvl3

Prototype:

```
uint32_t JumpHDPLvl3(uint32_t VectorTableAddr, uint32_t MPUIndex)
```

User code function call example:

```
RSSLIB_PFUNC->JumpHDPLvl3((uint32_t)NextVectorTableAddr, 1U);
```

Arguments:

- `VectorTableAddr`
  - Input parameter: Address of the next vector table to apply.
  - The vector table format follows the one used by the Cortex-M7 core.
- `MPUIndex`
  - Input parameter: MPU region index (index 0 to 15 allowed). The caller function must define but keep the corresponding MPU region disabled before calling JumpHDPLvl3. The function enables the MPU region before jumping to the reset handler of the vector table, which belongs to the MPU region.

The user calls `JumpHDPLvl3` to close the user flash HDPL1 and HDPL2 areas by incrementing HDPL up to 3, then jumps to the reset handler embedded within the vector table, whose address is passed as an input parameter.

After closing HDPL1 and HDPL2, `JumpHDPLvl3` enables the MPU region provided as an input parameter. Once the MPU is enabled, the function sets the new vector table, sets the SP to the address provided in this new vector table, and jumps to the reset handler function supported by it.

Returned values:

- On successful execution, the function does not return and does not push LR onto the stack.
- In the case of failure (invalid input parameter value), the `JumpHDPLvl3` function returns 0xF5F5F5F5.

GetProductState

Prototype:

```
uint32_t GetProductState(void)
```

User code function call example:

```
RSSLIB_PFUNC->GetProductState();
```

Arguments: None

The user calls GetProductState to reteive the STM32 product state.

Returned values:

- On successful call, the function returns one of the following product states:
  - 0x39: Open product state.
  - 0x17: Provisioning product state.
  - 0x72: Closed product state.
  - 0x5C: Locked product state.
- In case of failure (due to invalid product configuration), GetProductState returns 0xF5F5F5F5 (Bad state).

4 RAM ECC monitoring (RAMECC)

4.1 Introduction

STM32H7Pxxx devices feature a RAM error correction code monitoring unit (RAMECC). This provides a means for application software to enable, verify the ECC status, and execute service routines when an error occurs.

4.2 RAMECC main features

The internal SRAM can be protected by ECC when enabled. The ECC mechanism is based on the SECDED algorithm. It supports single- and double-error detection, as well as single-error correction:

- Seven ECC bits are added per 32-bit word.
- Eight ECC bits are added per 64-bit word.

RAM data word integrity is checked at each memory read access, or partial RAM word write operation. Two cycles are required to perform a partial RAM word write (read-modify-write).

The RAM ECC monitoring unit includes the following features:

- RAM ECC monitoring per domain
- RAM failing address/data identification
- Error code correction (ECC)

4.3 RAMECC functional description

ECC is supported by AXISRAM1-4, BKPSRAM, data and instruction tightly coupled memory (DTCM/ITCM). The ECC is enabled/disabled by the user option byte ECC_ON_SRAM in *Section 5.9.33: FLASH option byte word 1 status register (FLASH_OBW1SR)*.

Caution: Enabling ECC has a significant impact on performance. Check the device datasheet for the maximum CPU and bus frequencies that can be achieved with and without the ECC enabled.

As the ECC is calculated and checked for a 32/64-bit word, the byte and half-word write accesses are managed by the SRAM interface by first reading the whole word, then write the word again with the new byte/half-word value. ECC double errors are also detected during this byte or half-word AHB write accesses (read/modify/write done by interface). The byte or half-word write access latency is increased by 2 bus clock cycles.

Caution: In case of a byte or half-word write on SRAM with ECC, the read/modify/write operation is done in a buffer. The buffer content is written into the SRAM. The SRAM is then released. If a reset occurs during the operation, the SRAM content is not updated.

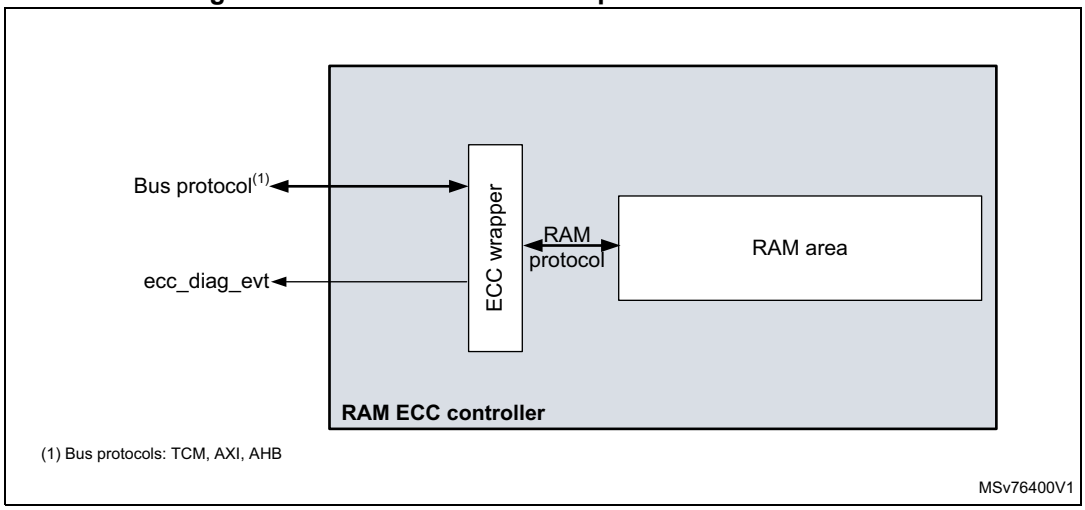
RAMECC block diagram

An ECC controller is associated with each RAM area. It performs the following functions:

- ECC encoding: ECC code computation and storage
- ECC decoding: RAM data word loading and ECC code decoding to detect errors
- Error detection: single- and double-error detection
- Error correction: single-error correction

The following figure describes the implementation of RAM ECC controllers.

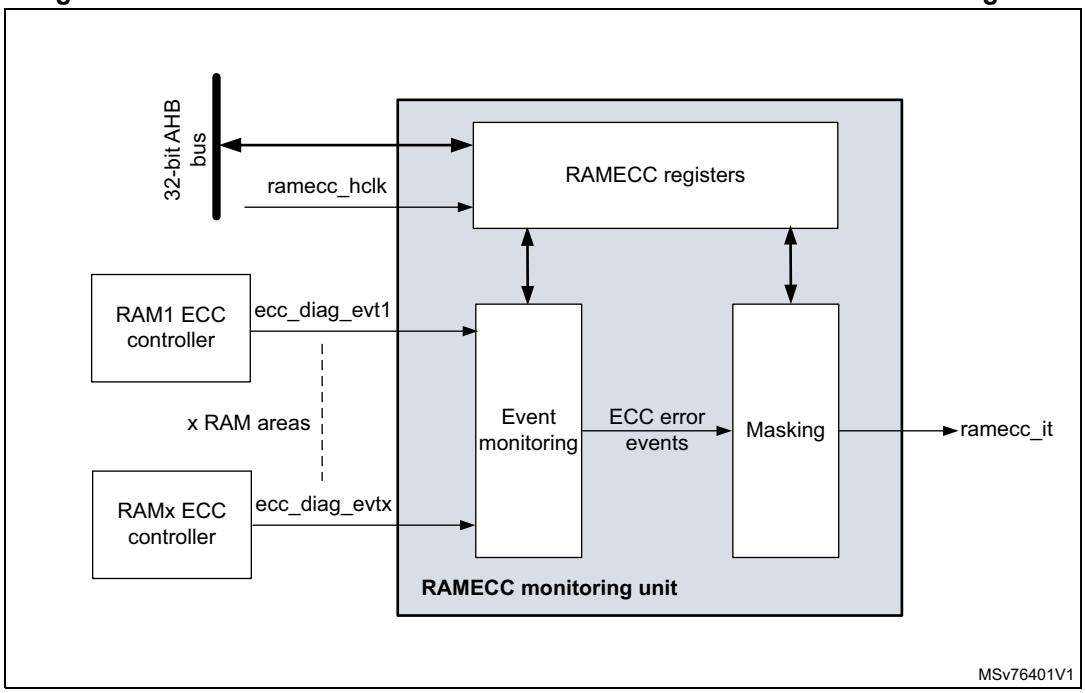
Figure 6. RAM ECC controller implementation schematic



A dedicated RAM ECC monitoring area is defined for each domain (see *Section 4.3.2: RAMECC monitor mapping*. The RAMECC allows the collection of ECC diagnostic events from each RAM ECC controller and provides a means for the CPU to verify the ECC status.

Figure 7 shows the connection schematic between the RAM ECC controller and the RAMECC monitoring unit.

Figure 7. Connection between RAM ECC controller and RAMECC monitoring unit



4.3.1 RAMECC internal signals

Table 17 lists the internal signals that control the RAMECC unit.

Table 17. RAMECC internal input/output signals

Internal signal name	Signal type	Description
ramecc_hclk	Input	AHB clock
ecc_diag_evtx	Input	ECC diagnostic event generated by RAMx ECC controller x
ramecc_it	Output	Interrupt generated by the RAMECC monitoring unit when an ECC error is detected.

4.3.2 RAMECC monitor mapping

STM32H7Pxxx devices feature two RAMECC units, RAMECC1 and RAMECC2. RAMECC1 monitors the AXI and TCM memories. RAMECC2 monitors the backup domain memory. The inputs from the ECC controllers are mapped to monitors as described in *Table 18*. The RAM ECC event monitoring status and configuration registers are described in *Section 4.6: RAMECC registers*.

Table 18. ECC controller mapping

RAMECC unit(1)	Monitor number	Description
RAMECC1	0	AXISRAM1 ECC controller monitoring unit
	1	AXISRAM2 ECC controller monitoring unit
	2	AXISRAM3 ECC controller monitoring unit
	3	AXISRAM4 ECC controller monitoring unit
	4	ITCM-RAM ECC controller monitoring unit
	5	DTCM0-RAM ECC controller monitoring unit
	6	DTCM1-RAM ECC controller monitoring unit
RAMECC2	0	Reserved
	1	BKPRAM ECC controller monitoring unit

1. See *Table 7: Peripheral boundary addresses*.

Single and double ECC errors

When a single error is detected, it is automatically corrected and the SEDCF bit is set in the *RAMECC monitor x status register (RAMECC_MxSR)*. When a double error is detected, the DEDF bit, or the DEBWDF bit in the case of a byte write, is set in the *RAMECC monitor x status register (RAMECC_MxSR)*.

For single and double errors:

- An interrupt is generated if enabled by the appropriate bit in the *RAMECC interrupt enable register (RAMECC_IER)*. Errors cannot be detected while the corresponding flag bit is set.
- The context (failing address, data, ECC code) is stored in the
  - *RAMECC monitor x failing address register (RAMECC_MxFAR)*
  - *RAMECC monitor x failing data register (RAMECC_MxFDR)*
  - *RAMECC monitor x failing ECC error code register (RAMECC_MxFECR)* if enabled by the ECCELEN bit in the *RAMECC monitor x configuration register (RAMECC_MxCR)*

4.3.3 ECC fault injection test on SRAM1, SRAM3, and TCM

A fault injection mechanism allows testing of 1- or 2-bit errors per word (for single or double error test respectively).

- ECCTEA = 0: Normal mode, ECC, and data memory access granted
- ECCTEA = 1: ECC memory access gated, data memory access granted

For example:

Initialization

1. Set the normal mode: ECCTEA = 0
2. Write 0xCAFEDECA, ECC = 0x19 (computed by ECC module)

Data modification

3. Bloc ECC memory access: ECCTEA = 1
4. Grant data modification: Write 0xCAFEDECB (last bit flipped)
5. Return to normal mode: ECCTEA = 0
6. READ 0xCAFEDECB with ECC = 0x19 single error ='1' (if enabled a flag is set and an interrupt to the NVIC is generated)

An ECC fault can trigger a system break event in TIM1/TIM8/TIM15/TIM16/TIM17/TIM20 if one or more of the following bits is set in the SBS_BKLOCKR register:

- ARAM1ECC_BL
- ARAM2ECC_BL
- ARAM3ECC_BL
- ARAM4ECC_BL
- ITCMECC_BL
- DTCMECC_BL

4.3.4 Read access latency

To correctly read data from SRAMs, the number of wait states is automatically set to one when ECC is enabled

4.4 RAMECC low-power modes

Table 19. Effect of low-power modes on RAMECC

Mode	Description
Sleep	No effect. RAMECC interrupt causes the device to exit the Sleep mode.
Stop	The content of RAMECC registers is kept. The ECC is functional and ECC error interrupt causes the device to exit from Stop mode.
Standby	The RAMECC peripheral is powered down and must be reinitialized after exiting Standby mode.

4.5 RAMECC interrupts

Table 20 lists the RAMECC interrupt requests.

Table 20. RAMECC interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit the Sleep/Stop mode	Exit the Standby mode
RAMECC	ECC single error detection and correction	SEDCF	GECCSEIE ECCSEIE GIE	Write 0 in SEDCF	Yes	No
	ECC double error detection	DEDF	GECCDEIE ECCDEIE GIE	Write 0 in DEDF	Yes	No
	ECC double error on byte write detection	DEBWDF	GECCDEBWIE ECCDEBWIE GIE	Write 0 in DEBWF	Yes	No

4.6 RAMECC registers

RAMECC registers can be accessed only in 32-bit (word) mode. Byte and half-word formats are not allowed.

4.6.1 RAMECC interrupt enable register (RAMECC_IER)

Address offset: 0x00

Reset value: 0x0000 0000

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

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

Bit 3 **GECCDEBWIE**: Global ECC double error on byte write (BW) interrupt enable
When GECCDEBWIE bit is set to 1, an interrupt is generated when an ECC double detection error occurs during a byte write operation to RAM (incomplete word write).
0: No interrupt generated when an ECC double detection error occurs on byte write
1: Interrupt generated if an ECC double detection error occurs on byte write

Bit 2 **GECCDEIE**: Global ECC double error interrupt enable
When GECCDEIE bit is set to 1, an interrupt is generated when an ECC double detection error occurs during a read operation from RAM.
0: No interrupt generated when an ECC double detection error occurs
1: Interrupt generated if an ECC double detection error occurs

Bit 1 **GECCSEIE**: Global ECC single error interrupt enable
When GECCSEIE bit is set to 1, an interrupt is generated when an ECC single error occurs during a read operation from RAM.
0: No interrupt generated when an ECC single error occurs
1: Interrupt generated when an ECC single error occurs

Bit 0 **GIE**: Global interrupt enable
When GIE bit is set to 1, an interrupt is generated when any ECC error occurs.
0: No interrupt generated when an ECC error occurs
1: Interrupt generated when an ECC error occurs

4.6.2 RAMECC monitor x configuration register (RAMECC_MxCR)

Address offset: 0x00 + 0x20 * (x+1), (x = 0 to 6)

Reset value: 0x0000 0000

x is the ECC monitoring unit number - see *Table 18: ECC controller mapping*

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

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **ECCTEA:** ECC test ECC access

This bit is used to inject a fault during ECC operation testing. This bit has no effect in RAMECC2.

0: Inactive

1: Write and read access blocked on ECC memory

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **ECCELEN**: ECC error latching enable

When ECCELEN bit is set to 1, if an ECC error occurs (both for single error correction or double detection) during a read operation, the context (address, data and ECC code) that generated the error are latched to their respective registers.

0: No error context preserved when an ECC error occurs

1: Error context preserved when an ECC error occurs

Bit 4 **ECCDEBWIE:** ECC double error on byte write (BW) interrupt enable

When ECCDEBWIE bit is set to 1, monitor x generates an interrupt when an ECC double detection error occurs during a byte write operation to RAM.

0: No interrupt generated when an ECC double detection error occurs on byte write

1: Interrupt generated if an ECC double detection error occurs on byte write

Bit 3 **ECCDEIE:** ECC double error interrupt enable

When ECCDEIE bit is set to 1, monitor x generates an interrupt when an ECC double detection error occurs during a read operation from RAM.

0: No interrupt generated when an ECC double detection error occurs

1: Interrupt generated if an ECC double detection error occurs

Bit 2 **ECCSEIE**: ECC single error interrupt enable

When ECCSEIE bit is set to 1, monitor x generates an interrupt when an ECC single error occurs during a read operation from RAM.

0: No interrupt generated when an ECC single error occurs

1: Interrupt generated when an ECC single error occurs

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

4.6.3 RAMECC monitor x status register (RAMECC_MxSR)

Address offset: 0x24 + 0x20 * x, (x = 0 to 6)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEBWDF	DEDF	SEDCF
													rc_w0	rc_w0	rc_w0

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

Bit 2 **DEBWDF:** ECC double error on byte write (BW) detected flag
This bit is set by hardware. It is cleared by software by writing a 0
0: No error detected
1: Error detected

Bit 1 **DEDF:** ECC double error detected flag
This bit is set by hardware. It is cleared by software by writing a 0
0: No error detected
1: Error detected

Bit 0 **SEDCF:** ECC single error detected and corrected flag
This bit is set by hardware. It is cleared by software by writing a 0
0: No error detected and corrected
1: Error detected and corrected

4.6.4 RAMECC monitor x failing address register (RAMECC_MxFAR)

Address offset: 0x28 + 0x20 * x, (x = 0 to 6)

Reset value: 0x0000 0000

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

Bits 31:0 **FADD[31:0]:** ECC error failing address
When an ECC error occurs the FADD bit field contains the address that generated the ECC error.

4.6.5 RAMECC monitor x failing data register (RAMECC_MxFDR)

Address offset: 0x2C + 0x20 * x, (x = 0 to 6)

Reset value: 0x0000 0000

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

Bits 31:0 **FDATA[31:0]:** Failing data

When an ECC error occurs the FDATA bit field contains the data that generated the error.
This bitfield is valid only in RAMECC2. It is reserved in RAMECC1.

4.6.6 RAMECC monitor x failing ECC error code register (RAMECC_MxFECR)

Address offset: 0x34 + 0x20 * x, (x = 0 to 6)

Reset value: 0x0000 0000

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

Bits 31:0 **FEC[31:0]:** Failing error code

When an ECC error occurs the FEC bit field contains the ECC failing code that generated the error.
This bitfield is valid only in RAMECC2. It is reserved in RAMECC1.

4.6.7 RAMECC register map

Table 21. RAMECC register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**RAMECC_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GECCDEBWIE	GECCDEIE	GECCSEIE	GIE
	Reset value																													0	0	0	0

Table 21. RAMECC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	**RAMECC_M0CR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCTEA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCELEN	ECCDEBWIE	ECCDEIE	ECCSEIE	Res.	Res.
	Reset value																0											0	0	0	0		
0x24	**RAMECC_M0SR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEBWDF	DEDF	SEDCF
	Reset value																														0	0	0
0x28	**RAMECC_M0FAR**(1)	FADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**RAMECC_M0FDR**(1)	FDATA[31:0]]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved	Res.																															
0x34	**RAMECC M0FECR**(1)	FEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 to 0x3C	Reserved	Res.																															
0x40	**RAMECC_M1CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCTEA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCELEN	ECCDEBWIE	ECCDEIE	ECCSEIE	Res.	Res.
	Reset value																0											0	0	0	0		
0x44	**RAMECC_M1SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEBWDF	DEDF	SEDCF
	Reset value																														0	0	0
0x48	**RAMECC_M1FAR**	FADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	**RAMECC_M1FDR**	FDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	Reserved	Res.																															
0x54	**RAMECC_ M1FECR**	FEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x58 to 0x5C	Reserved	Res.																															
0x60	**RAMECC_M2CR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCTEA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCELEN	ECCDEBWIE	ECCDEIE	ECCSEIE	Res.	Res.
	Reset value																0											0	0	0	0		
0x64	**RAMECC_M2SR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEBWDF	DEDF	SEDCF
	Reset value																														0	0	0
0x68	**RAMECC M2FAR**(1)	FADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6C	**RAMECC M2FDR**(1)	FDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21. RAMECC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x70	Reserved	Res.																															
0x74	**RAMECC_M2FECR**[1]	FEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x78 to 0x7C	Reserved	Res.																															
0x80	**RAMECC_M3CR**[1]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCTEA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCELEN	ECCDEBWIE	ECCDEIE	ECCSEIE	Res.	Res.
	Reset value																0											0	0	0	0		
0x84	**RAMECC_M3SR**[1]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEBWDF	DEDF	SEDCF
	Reset value																														0	0	0
0x88	**RAMECC_M3FAR**[1]	FADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C	**RAMECC_M3FDR**[1]	FDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x90	Reserved	Res.																															
0x94	**RAMECC_M3FECR**[1]	FEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98 to 0x9C	Reserved	Res.																															
0xA0	**RAMECC_M4CR**[1]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCTEA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCELEN	ECCDEBWIE	ECCDEIE	ECCSEIE	Res.	Res.
	Reset value																0											0	0	0	0		
0xA4	**RAMECC_M4SR**[1]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEBWDF	DEDF	SEDCF
	Reset value																														0	0	0
0xA8	**RAMECC_M4FAR**[1]	FADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xAC	**RAMECC_M4FDR**[1]	FDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB0	Reserved	Res.																															
0xB4	**RAMECC_M4FECR**[1]	FEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB8 to 0xBC	Reserved	Res.																															
0xC0	**RAMECC_M5CR**[1]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCTEA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCELEN	ECCDEBWIE	ECCDEIE	ECCSEIE	Res.	Res.
	Reset value																0											0	0	0	0		

Table 21. RAMECC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xC4	**RAMECC_M5SR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEBWDF	DEDF	SEDCF
	Reset value																														0	0	0
0xC8	**RAMECC_M5FAR**(1)	FADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xCC	**RAMECC_M5FDR**(1)	FDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD0	Reserved	Res.																															
0xD4	**RAMECC_M5FECR**(1)	FEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD8 to 0xDC	Reserved	Res.																															
0xE0	**RAMECC_M6CR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCTEA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCELEN	ECCDEBWIE	ECCDEIE	ECCSEIE	Res.	Res.
	Reset value																0											0	0	0	0		
0xE4	**RAMECC_M6SR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEBWDF	DEDF	SEDCF
	Reset value																														0	0	0
0xE8	**RAMECC_M6FAR**(1)	FADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xEC	**RAMECC_M6FDR**(1)	FDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF0	Reserved	Res.																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF4	**RAMECC_M6FECR**(1)	FEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF8 to 0xFC	Reserved	Res.																															

1. RAMECC1 only. Reserved in RAMECC2.

5 Embedded flash memory (FLASH)

5.1 Introduction

The embedded flash memory (FLASH) manages the accesses to 2 Mbytes of embedded nonvolatile memory (NVM). It implements the read, program and erase operations, error corrections as well as various integrity and confidentiality protection mechanisms.

The embedded flash memory manages the automatic loading of nonvolatile user option bytes at power-on reset, and implements the dynamic update of those options.

5.2 FLASH main features

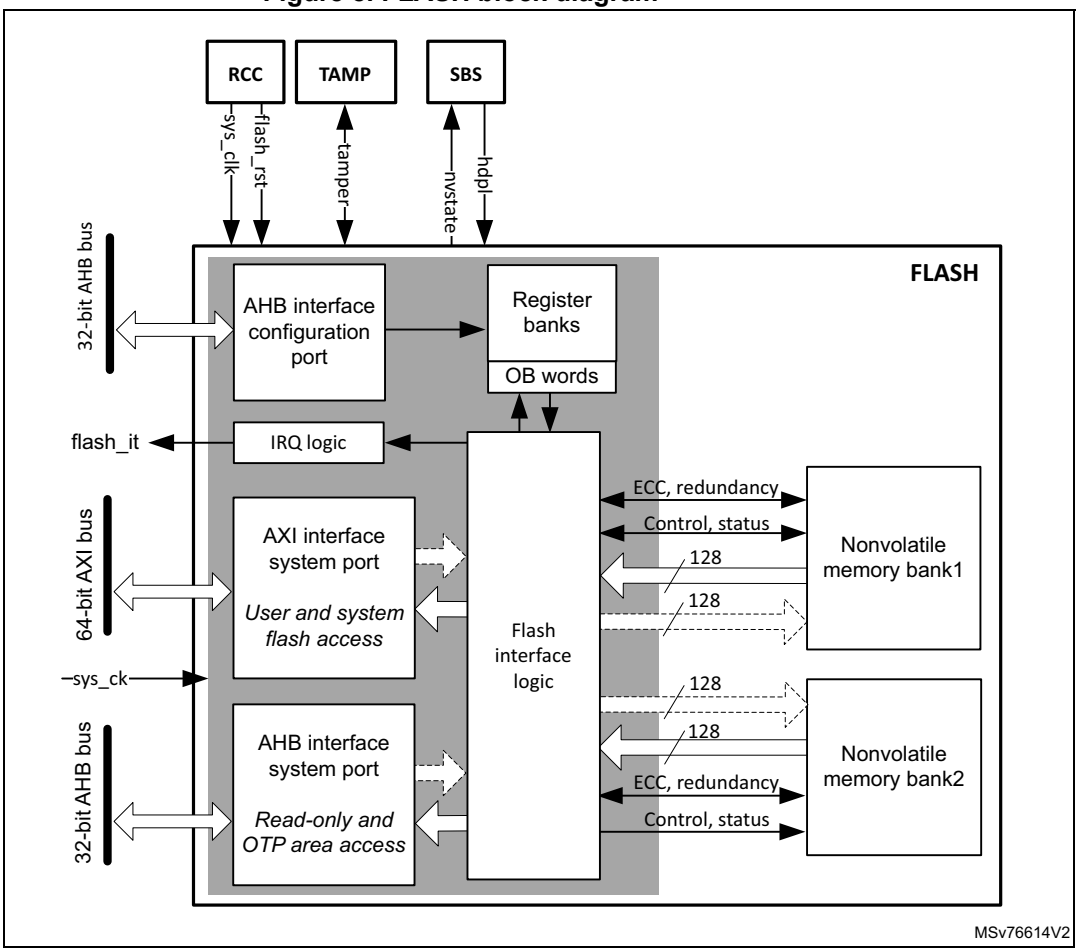
- *(STM32H7PxxI devices only)* 2 Mbytes of nonvolatile user flash memory divided into two physical banks of 1 Mbyte each
- *(STM32H7PxxG devices only)* 1 Mbytes of nonvolatile user flash memory divided into two physical banks of 512 Kbytes each
- Flash memory read operations supporting multiple length (64 bits, 32 bits, 16 bits, or one byte)
- Flash memory programming by 128 bits (user area) and 16 bits (OTP area)
- 8-Kbyte page erase, and bank erase (conditioned)
- Dual-bank organization supporting:
  - Simultaneous operations: two read/program/erase operations executed in parallel on both banks
  - Bank swapping: physical bank allocation to user flash memory bank can be changed with option bytes, along with the corresponding registers. Security flags (write protection, HDPL) remain valid for the physical bank, so the data are not reveled by swapping.
- Error Code Correction (ECC): one error detection/correction or two error detections per 128-bit flash word using 9 ECC bits
- Cyclic redundancy check (CRC) hardware module
- User configurable nonvolatile option byte words
- Nonvolatile security life-cycle management of the device
  - Includes 128-bit password-based regression with RAM and flash automatic erase
- Temporal isolation enforcement based on application HDPL stored in SBS register
  - Securing one user flash area per bank, expandable using a volatile extension
  - Unique boot entry enforcement on a closed device
  - Dedicated nonvolatile option byte keys storage (32x 128-bit keys available for HDP levels 0 to 2)
- Page group write-protection (WRPG) per bank, protecting up to 32 group of 4 pages (32 Kbytes) per bank
- 512 words of 16-bit of one-time programmable (OTP) reserved for user information
- 512-byte read-only area reserved to STMicroelectronics information
- Read command queue to streamline flash operations

5.3 FLASH functional description

5.3.1 FLASH block diagram

Figure 8 shows the embedded flash memory block diagram, with its interaction with the rest of the device.

Figure 8. FLASH block diagram



For more information on RCC, *SBS* and *TAMP*, refer to the respective sections.

5.3.2 FLASH internal signals

Table 22 describes a list of the useful to know internal signals available at embedded flash memory level. These signals are not available on the microcontroller pads.

Table 22. FLASH internal input/output signals

Signal name	Signal type (1)	Description
sys_rst	Input	System reset of the device
flash_rst		Device power on reset triggering an option byte loading (see *Section 5.3.6*)
sys_ck		System bus clock
flash_it	Output	Embedded flash memory interrupt request
hdpl[7:0]	Input	Hide protection level stored in SBS_HDPLSR register of SBS peripheral. Used by embedded flash memory.
nvstate[7:0]	Output	Nonvolatile security state stored in FLASH_NVSR register. Used by SBS peripheral
tamper	Input, output	Internal signals managing tamper events in the device (see *Section 5.3.16*).

1. Only digital signals

5.3.3 FLASH architecture and integration in the system

The embedded flash memory is a central resource for the whole microcontroller. It serves as an interface to two nonvolatile memory banks, and organizes the memory in a very specific way. The embedded flash memory also proposes a set of security features to protect the assets stored in the nonvolatile memory at boot time, at runtime and during firmware and configuration upgrades.

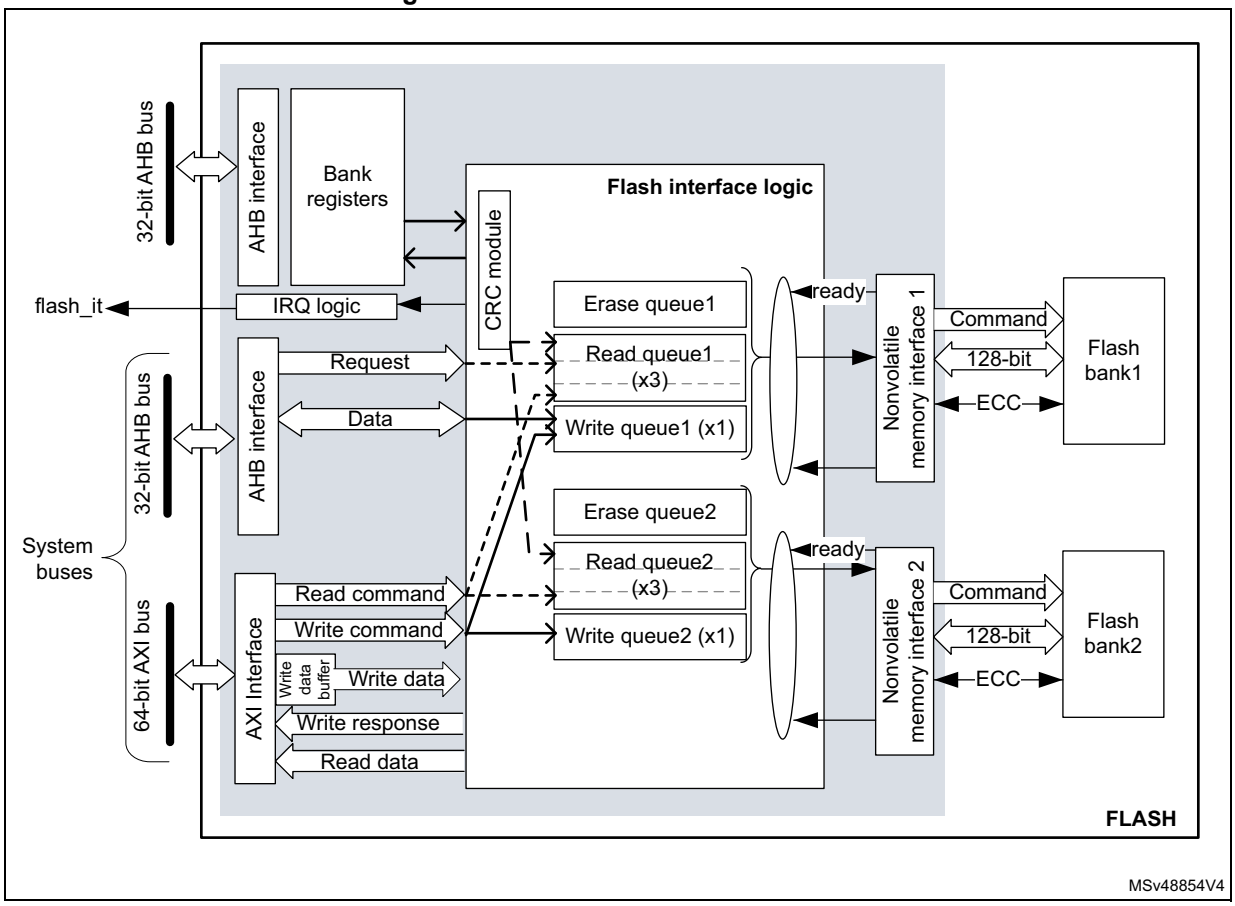
The embedded flash memory offers three different interconnect interfaces:

- An AXI 64-bit slave port to access code/data
- An AHB 32-bit system slave port to access read-only and OTP area
- An AHB 32-bit configuration slave port to access the register bank

Note: The application can simultaneously request a read and a write operation through the AXI slave port.

The embedded flash memory micro-architecture is shown in *Figure 9*.

Figure 9. Detailed FLASH architecture



Behind the system interfaces, the embedded flash memory implements various command queues and buffers to perform flash read, write and erase operations with maximum efficiency.

By the addition of a read and write data buffer, the AXI slave port handles the following access types:

- Multiple lengths: 64 bits, 32 bits, 16 bits and 8 bits
- Single or burst accesses
- Write wrap burst must not cross 16-byte aligned address boundaries to target exactly one flash word

The AHB system slave port supports only 16-bit or 32-bit word accesses. For 8-bit accesses, an AHB bus error is generated and write accesses are ignored.

The AHB configuration slave port supports 8-bit, 16-bit and 32-bit word accesses.

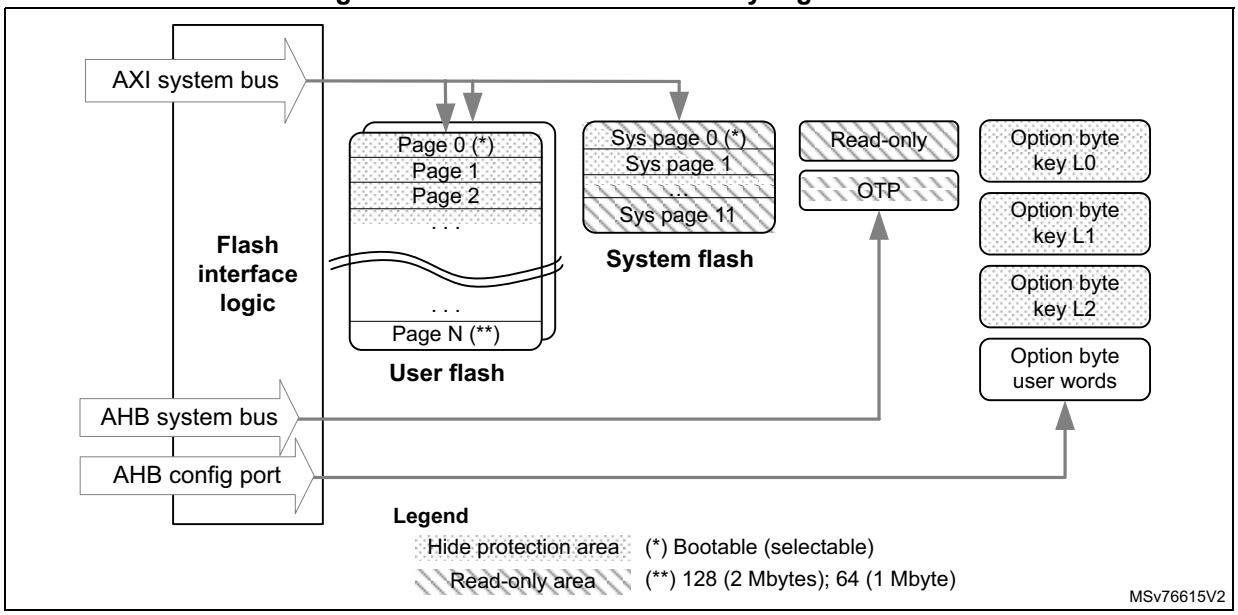
The embedded flash memory is built in such a way that only one read or write operation can be executed at a time on a given bank.

5.3.4 Flash memory architecture and usage

Flash memory architecture

Figure 10 shows the nonvolatile memory organization in the embedded flash memory.

Figure 10. Embedded flash memory organization



The embedded flash nonvolatile memory is composed of:

- A main memory block, organized as two banks divided in pages of 8 Kbytes each. It features flash-word rows of 128 bits + 9 bits of ECC per word.
  - *(STM32H7PxxI devices only)* memory size is 2-Mbyte, organized as two banks of 1-Mbyte each. Each bank is in turn divided in 128 pages.
  - *(STM32H7PxxG devices only)* memory size is 1-Mbyte, organized as two banks of 512-Kbyte each. Each bank is in turn divided in 64 pages.
- A 128 Kbytes system memory block, divided into two 64-Kbyte banks. Each bank is in turn divided into eight 8-Kbyte pages. The system flash memory is ECC protected.
- A set of nonvolatile user option byte words, loaded at reset by the embedded flash memory and accessible by the application software only through the AHB configuration register interface.
- A 1 Kbyte one-time-programmable (OTP) area that can be written only once by the application software.
- A 512-byte read-only area written by STMicroelectronics during SoC manufacturing.
- 32 key slots of 16-byte per HDP level (up to level 2, included), loaded by software through the AHB configuration register interface. All the option byte keys benefit from the device hide protection mechanism.

Note: The system flash memory cannot be programmed nor erased by any application.

Since the OTP area has no specific protection, it must not be used to store confidential information.

The overall flash memory architecture and its corresponding access interface is summarized in *Table 23 (STM32H7PxxI devices only)* and *Table 24 (STM32H7PxxG devices only)*.

Table 23. Flash memory organization (2 Mbytes)

Flash memory area		Address range	Size (bytes)	ECC type	Region name	Access interface
User main memory	Bank A	0x0800 0000-0x0800 1FFF	8 K	128+ 9bit	user page 0	AXI port
		0x0800 2000-0x0800 3FFF	8 K		user page 1	
		...	...		...	
		0x080F E000-0x080F FFFF	8 K		user page 127	
	Bank B	0x0810 0000-0x0810 1FFF	8 K	128+ 9bit	user page 0	AXI port
		0x0810 2000-0x0810 3FFF	8 K		user page 1	
		...	...		...	
		0x081F E000-0x081F FFFF	8 K		user page 127	
System memory(1)	Bank A	0x1FF0 0000-0x1FF0 1FFF	8 K	128+ 9bit	system page 0	AXI port
		...	...		...	
		0x1FF0 E000-0x1FF0 FFFF	8 K		system page 7	
	Bank B	0x1FF1 0000-0x1FF1 1FFF	8 K	128+ 9bit	system page 0	AXI port
		...	...		...	
		0x1FF1 E000-0x1FF1 FFFF	8 K		system page 7	
Special region		0x08FF F000-0x08FF F3FF	1 K	-	OTP area	AHB system port
		0x08FF F400-0x08FF F7FF	1 K	-	Reserved	
		0x08FF F800-0x08FF F9FF	512	-	Read-only area	
Option bytes		N/A	-	16+ 6bit	user option bytes	Registers only

1. Cannot be erased by application software.

Table 24. Flash memory organization (1 Mbyte)

Flash memory area		Address range	Size (bytes)	ECC type	Region name	Access interface
User main memory	Bank A	0x0800 0000-0x0800 1FFF	8 K	128+ 9bit	user page 0	AXI port
		0x0800 2000-0x0800 3FFF	8 K		user page 1	
		...	...		...	
		0x0807 E000-0x0807 FFFF	8 K		user page 63	
	Bank B	0x0808 0000-0x0808 1FFF	8 K	128+ 9bit	user page 64	AXI port
		0x0808 2000-0x0808 3FFF	8 K		user page 65	
		...	...		...	
		0x080F E000-0x080F FFFF	8 K		user page 127	
System memory(1)	Bank A	0x1FF0 0000-0x1FF0 1FFF	8 K	128+ 9bit	system page 0	AXI port
		...	...		...	
		0x1FF0 E000-0x1FF0 FFFF	8 K		system page 7	
	Bank B	0x1FF1 0000-0x1FF1 1FFF	8 K	128+ 9bit	system page 0	AXI port
		...	...		...	
		0x1FF1 E000-0x1FF1 FFFF	8 K		system page 7	
Special region		0x08FF F000-0x08FF F3FF	1 K	-	OTP area	AHB system port
		0x08FF F400-0x08FF F7FF	1 K	-	Reserved	
		0x08FF F800-0x08FF F9FF	512	-	Read-only area	
Option bytes		N/A	-	16+ 6bit	user option bytes	Registers only

1. Cannot be erased by application software.

Partition usage

In the life of the product-embedded flash memory stores different kinds of data. Data availability also varies according to the hide protection level (HDPL) selected by the application at any given time. See *Section 10: System configuration, boot and security (SBS)* for more details.

The typical flash usages are described on *Figure 11*.

Figure 11. Embedded flash memory usage

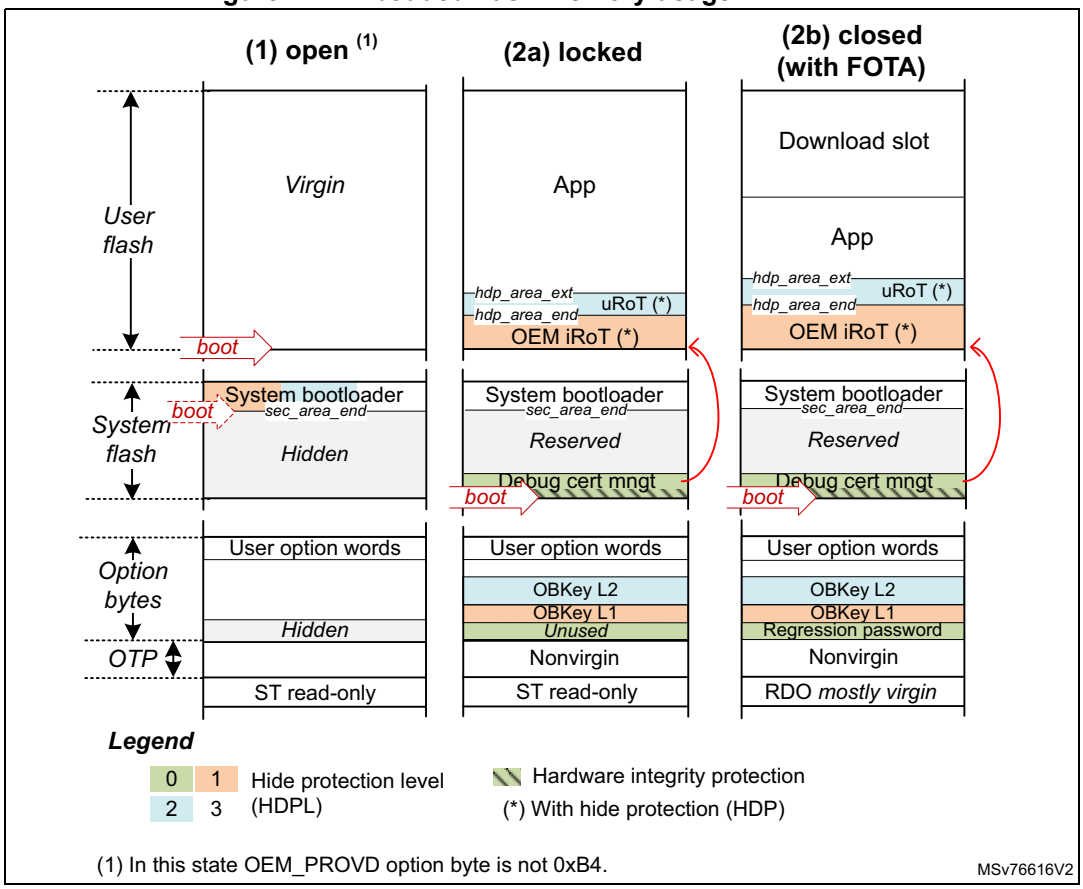
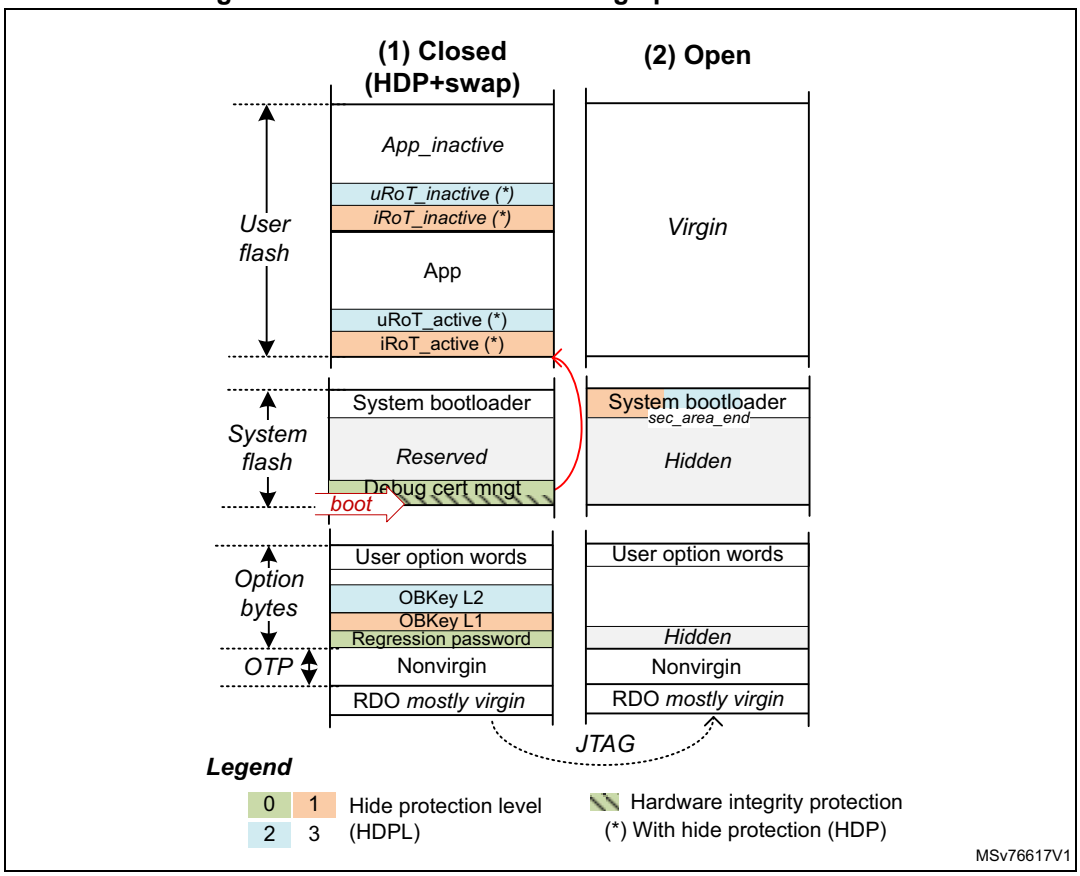


Figure 12 shows how the embedded flash memory is used when an OEM returns a provisioned production device to an open state (for example when returning a part for field return analysis).

Note: In any device, if the application knows the decommissioning key it can open the device using STM32 tools.

Figure 12. Flash decommissioning options



Bank swapping

The embedded flash memory offers a bank swapping feature that can be configured through the SWAP_BANK bit, always available for the application. Bank swap feature can be revoked using the BOOT_LOCK option.

For more information, refer to *Section 5.3.14: FLASH bank and register swapping*.

5.3.5 Flash hide protection schemes

Figure 10 gives an overview of the protection mechanism supported by the embedded flash memory:

- A user flash area can be defined as a user HDP area, with a hardware defined access policy based on STM32 hide protection level (HDPL).
- A system flash starting from page zero is defined as system HDP area, also with a hardware-defined access policy based on STM32 hide protection level (HDPL).
- Three sets of option byte keys are each reserved to a specific hide protection level.

For more information on both HDP areas, refer to *Section 5.5.4: Hide protection (HDP)*. More details on option byte keys can be found in *Section 5.4.4: Option byte key management*.

5.3.6 Overview of FLASH operations

Read operations

The embedded flash memory can perform read operations on the whole nonvolatile memory using various granularities: 64 bits, 32 bits, 16 bits or one byte. User and system flash memories are read through the AXI interface, while the option bytes are read through the register interface. Read-only and OTP special region are read through the system AHB interface.

The embedded flash memory supports read-while-write operations provided the read and write operations target different physical banks. Similarly read-while-read operations are supported when two read operations target different banks.

To increase efficiency, the embedded flash memory implements the buffering of consecutive read requests in the same physical bank.

For more details on read operations, refer to *Section 5.3.7: FLASH read operations*.

Program/erase operations

The embedded flash memory supports the following program and erase operations:

- Single flash word write (128-bit granularity for user flash memory, 16-bit granularity for OTP area), with the possibility for the application to force-write a user flash word with less than 128 bits in user flash memory
- Single page erase
- Bank erase
- Option byte words update (if allowed)

Thanks to its dual bank architecture, the embedded flash memory can perform any of the above write or erase operation on one physical bank while a read or another program/erase operation is executed on the other bank.

Note: Program and erase operations are subject to the various protections that could be set on the embedded flash memory (see next).

For more details, refer to *Section 5.3.8: FLASH program operations* and *Section 5.3.9: FLASH erase operations*.

Protection mechanisms

The embedded flash memory supports different protection mechanisms:

- Configuration protection
- Write protection
- Temporal isolation enforcement of nonvolatile secrets (code/data, keys), based on the hide protection level information stored in SBS_HDPLSR register.
- Life-cycle management, based on flash root of trust option bytes
- OTP locking

For more details refer to *Section 5.5: FLASH protection mechanisms*.

Option byte loading

As part of the life-cycle management, the embedded flash memory reliably loads the nonvolatile option bytes stored in nonvolatile memory after every power on reset, enforcing boot and security options to the whole device when the embedded flash memory becomes functional again. For more details refer to *Section 5.4: FLASH option bytes*.

5.3.7 FLASH read operations

Read operation overview

The embedded flash memory supports, for each memory bank, the execution of one read command while two are waiting in the read command queue. Multiple read access types are also supported as defined in *Section 5.3.3: FLASH architecture and integration in the system*.

The read commands to each physical bank are associated with a 128-bit read data buffer. These commands can be issued either by the AHB (OTP or read-only areas) or by the AXI interface (user flash or system flash memory).

Note: The embedded flash memory can perform single error correction and double error detection while read operations are being executed (see Section 5.3.10: FLASH parallel operations).

The AXI interface read channel operates as follows:

- When the read command queue is full, any new AXI read request stalls the bus read channel interface and consequently the master that issued that request.
- If several consecutive read accesses request data that belong to the same flash data word (128 bits), the data are read directly from the current data read buffer, without triggering additional flash read operations. This mechanism occurs each time a read access is granted. When a read access is rejected for security reasons (e.g. HDP protected word), the corresponding read error response is issued by the embedded flash memory and no read operation to flash memory is triggered.

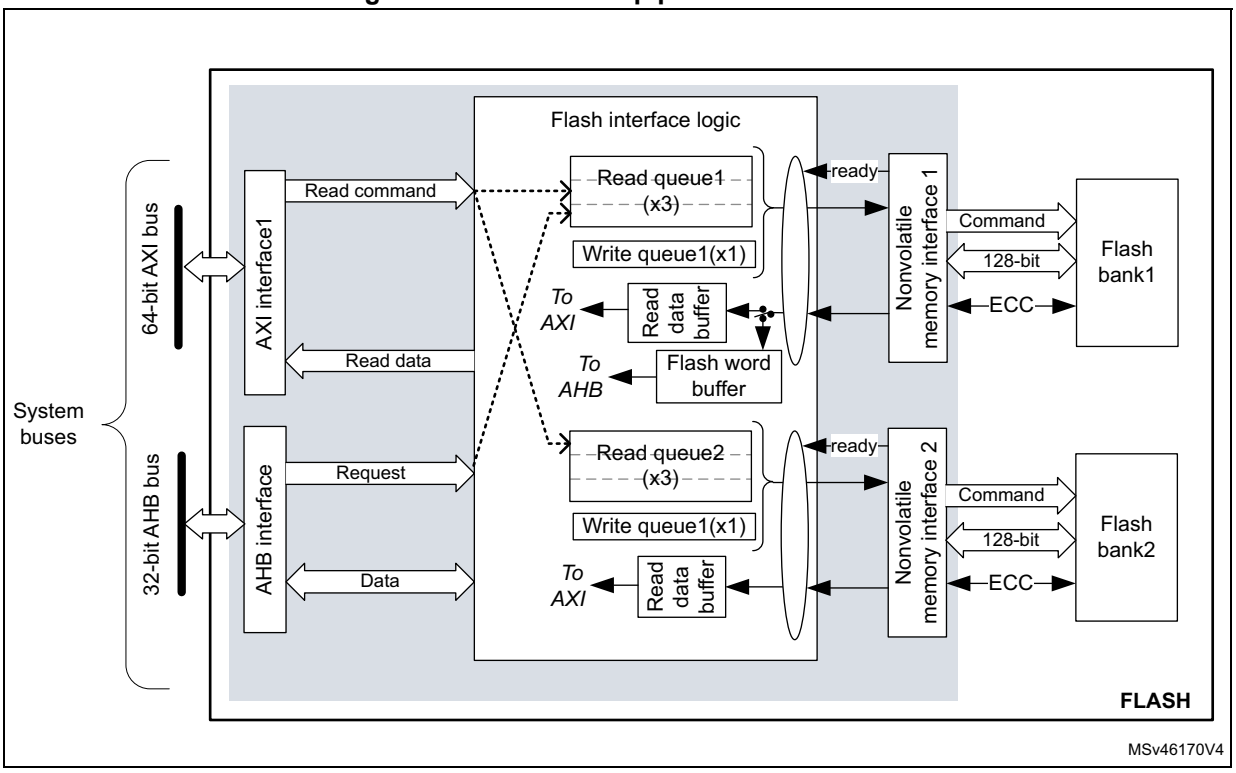
The AHB system interface operates as follows:

- Until the AHB read request has been served, the embedded flash memory stalls the AHB bus and consequently the master that issued that request.

The Read pipeline architecture is summarized in *Figure 13*.

For more information on bus interfaces, refer to *Section 5.3.3: FLASH architecture and integration in the system*.

Figure 13. FLASH read pipeline architecture



Single read sequence

The recommended simple read sequence is the following:

1. For AXI interface: Freely perform read accesses to any AXI-mapped area. For AHB interface: perform either 16-bit or 32-bit read accesses to the AHB-mapped area (byte accesses generate a bus error).
2. The embedded flash memory effectively executes the read operation from the read command queue buffer as soon as the nonvolatile memory is ready and the previously requested operations on this specific bank have been served.

Note: When reading an OTP data that has not been previously written, a double ECC error is reported and only 1's are returned (see Section 5.3.12 for details).

Adjusting read timing constraints

The embedded flash memory clock must be enabled and running before reading data from nonvolatile memory.

To correctly read data from flash memory, the number of wait states (LATENCY) must be correctly programmed in the flash access control register (FLASH_ACCR) according to the embedded flash memory AXI interface clock frequency (rcc_aclk) and the core voltage range of the device (V_{core}).

Table 25 shows the correspondence between the number of wait states (LATENCY), the programming delay parameter (WRHIGHFREQ), the embedded flash memory clock frequency and its supply voltage ranges.

Table 25. FLASH recommended number of wait states and programming delays

Number of wait states (LATENCY)	Programming delay (WRHIGHFREQ)	AXI interface clock maximum frequency versus V_{CORE} range		
		VOS2 range (1.05 V - 1.15 V)	VOS1 range (1.15 V - 1.26 V)	VOS0 range (1.26 V - 1.35 V)
0 WS (1 FLASH clock cycle)	00	40 MHz		
1 WS (2 FLASH clock cycles)		80 MHz		
2 WS (3 FLASH clock cycles)	01	120 MHz		
3 WS (4 FLASH clock cycles)		150 MHz	160 MHz	
4 WS (5 FLASH clock cycles)	10	N/A	200 MHz	
5 WS (6 FLASH clock cycles)		N/A	217 MHz	240 MHz
6 WS (7 FLASH clock cycles)	11	N/A	N/A	260 MHz

Adjusting the system frequency

After power-on, the embedded flash memory is clocked by the 64 MHz high-speed internal oscillator (HSI), with a voltage range set at VOS1.

When changing the AXI bus frequency, the application software must follow the below sequence in order to tune the number of wait states required to access the nonvolatile memory.

To increase the embedded flash memory clock source frequency:

1. If necessary, program the LATENCY and WRHIGHFREQ bits to the right value in the FLASH_ACCR register, as described in *Table 25*.
2. Check that the new number of wait states is taken into account by reading back the FLASH_ACCR register.
3. Modify the embedded flash memory clock source and/or the AXI bus clock prescaler in the RCC_CFGR register of the reset and clock controller (RCC).
4. Check that the new embedded flash memory clock source and/or the new AXI bus clock prescaler value are taken in account by reading back the embedded flash memory clock source status and/or the AXI bus prescaler value in the RCC_CFGR register of the reset and clock controller (RCC).

To decrease the embedded flash memory clock source frequency:

1. Modify the embedded flash memory clock source and/or the AXI bus clock prescaler in the RCC_CFGR register of reset and clock controller (RCC).
2. Check that the embedded flash memory new clock source and/or the new AXI bus clock prescaler value are taken into account by reading back the embedded flash memory clock source status and/or the AXI interface prescaler value in the RCC_CFGR register of reset and clock controller (RCC).
3. If necessary, program the LATENCY and WRHIGHFREQ bits to the right value in the FLASH_ACCR register, as described in *Table 25*.
4. Check that the new number of wait states has been taken into account by reading back the FLASH_ACCR register.

Read errors

The embedded flash memory embeds an error correction mechanism, as described in *Section 5.3.10*.

Single error correction and double error detection are performed for each read operation. In both cases the embedded flash memory reports read errors as described in *Section 5.7.5: Error correction code error (SNECCERRF/DBECCERRF)*.

Read interrupts

See *Section 5.8: FLASH interrupts* for details.

5.3.8 FLASH program operations

Program operation overview

The virgin state of each nonvolatile memory bitcell is 1. The embedded flash memory supports programming operations that can change any memory bitcell to 0. However these operations do not support the return of a bit to its virgin state. In this case an erase operation of the entire page is required.

Program operation consists in issuing write commands. The embedded flash memory supports the execution of one write command for each memory bank. For write accesses issued by the AXI interface, since a 9-bit ECC code is associated with each 128-bit data flash word, the embedded flash memory must always perform write operations to nonvolatile memory with a 128-bit word granularity.

Note: The application can decide to write as few as 8 bits to a 128-bit flash word. In this case, a force-write mechanism to the 128 bits + ECC is used (see FW bit of FLASH_CR register).

System flash memory cannot be written by the application software.

It is not recommended to overwrite a flash word that is not virgin. The result may lead to an inconsistent ECC code that is systematically reported by the embedded flash memory, as described in *Section 5.7.5: Error correction code error (SNECCERRF/DBECCERRF)*.

Write access requests issued by the AHB system interface are serialized with AXI commands and can only be used to program the memory OTP area. In this area, since a 6-bit ECC code is associated with each 16-bit data flash word, the embedded flash memory supports 16-bit or 32-bit write operations (8-bit write operations are not supported).

Note: The OTP area is typically write-protected on the final product, as described in Section 5.3.12: FLASH one-time programmable (OTP) area.

Erase operations to the OTP area are not supported.

The AXI interface write channel operates as follows:

- A 128-bit write data buffer is associated with the AXI interface. It supports multiple write access types (64 bits, 32 bits, 16 bits and 8 bits).
- When the write queue is full, any new AXI write request stalls the bus write channel interface and consequently the master that issued that request.

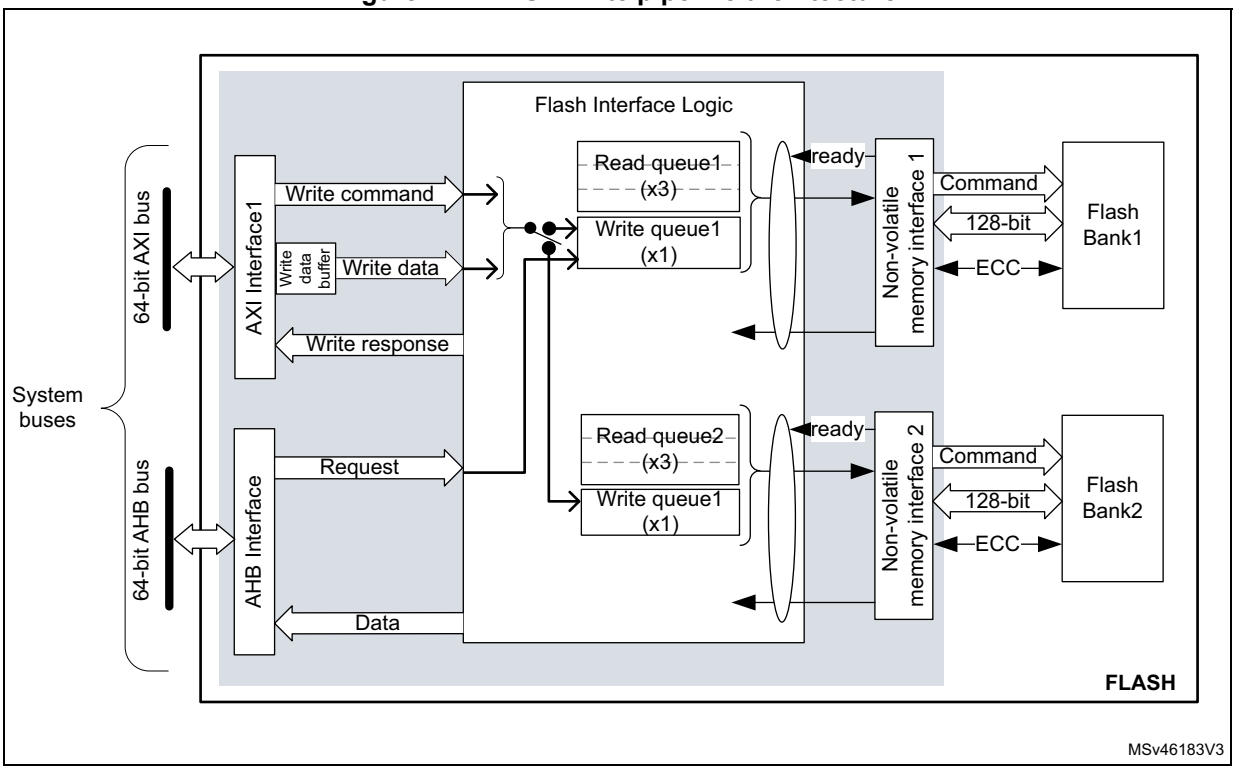
The AHB system interface operates as follows:

- Write commands issued by the AHB system interface are associated with a 137-bit flash word buffer. Byte accesses are not supported.
- When the write queue is full, any new AHB request stalls the bus interface and consequently the master that issued that request.

The write pipeline architecture is described in *Figure 14*.

For more information on bus interfaces, refer to *Section 5.3.3: FLASH architecture and integration in the system*.

Figure 14. FLASH write pipeline architecture



Managing write protections

Before programming a user page, the application software must check the protection of the targeted flash memory area.

The embedded flash memory checks the protection properties of the write transaction target at the output of the write queue buffer, just before the effective write operation to the nonvolatile memory:

- If a write protection violation is detected, the write operation is canceled and write protection error (WRPERRF) is raised in FLASH_xISR register (x = A or B).
- If the write operation is valid, the 9-bit ECC code is concatenated to the 128 bits of data and the write to nonvolatile memory is effectively executed.

Note: No write protection check is performed when the embedded flash memory accepts AXI write requests.

A similar mechanism exists for OTP areas with the following differences:

- If the write operation is valid, the 6-bit ECC code is concatenated to the 16 bits of data and the write to nonvolatile memory is effectively executed.

The write protection flag does not need to be cleared before performing a new programming operation.

Monitoring ongoing write operations

The application software can use three status flags located in FLASH_xSR in order to monitor ongoing write operations in the bank x. These status flags are available for each bank.

- **BUSY**: this bit indicates in real time that an effective write, erase or option byte change operation is ongoing to the nonvolatile memory. It is possible to know what type of operation is being executed polling the bits IS_PROGRAM, IS_ERASE and IS_OPTCHANGE in the same register.
- **QW**: this bit indicates that a write, erase or option byte change operation is pending in the write queue or command queue buffer. It remains high until the write operation is complete. It supersedes the BUSY status bit.
- **WBNE**: this bit indicates that the embedded flash memory is waiting for new data to complete the 128-bit write buffer. In this state the write buffer is not empty. It is reset as soon as the application software fills the write buffer, force-writes the operation using FW bit in FLASH_xCR, or disables all write operations in the bank x.

Enabling write operations

Before programming the user flash memory, the application software must make sure that PG bit is set in FLASH_xCR. If it is not set an unlock sequence must be used once (see *Section 5.5.1: FLASH configuration protection*) and the PG bit must be set.

When the option bytes need to be modified, the application software must make sure that FLASH_OPTCR is unlocked. If it is not the case, an unlock sequence must be used (see *Section 5.5.1: FLASH configuration protection*). For more information on option byte word modification, refer to *Section 5.4.3*.

Note: The application software must not unlock a register that is already unlocked, otherwise this register remains locked until next system reset.

If needed, the application software can update the programming delay as described at the end of this section.

Single write sequence

The recommended single write sequence in the bank x is the following:

1. Unlock the FLASH_xCR register, as described in *Section 5.5.1: FLASH configuration protection* (only if the register is not already unlocked).
2. Enable write operations by setting the PG bit in the FLASH_xCR register.
3. Check the protection of the targeted memory area.
4. Write one flash-word corresponding to 16-byte data starting at 16-byte aligned address.
5. Check that QW has been raised and wait until it is cleared in the FLASH_xSR register.

If step 4 is executed incrementally (for example byte per byte), the write buffer can become partially filled. In this case the application software can decide to force-write what is stored in the write buffer by using the FW bit in FLASH_xCR register. In this particular case, the unwritten bits are automatically set to 1. If no bit in the write buffer is cleared to 0, the FW bit has no effect.

Note: Using a force-write operation prevents the application from updating later the missing bits with a value different from 1, which is likely to lead to a permanent ECC error.

Any write access requested while the PG bit is cleared to 0 is rejected. In this case, no error is generated on the bus, but the PGSERRF flag is raised.

Clearing the programming sequence error (PGSERRF) and inconsistency error (INCERRF) is mandatory before attempting a write operation (see Section 5.7: FLASH error management for details).

Adjusting programming timing constraints

Program operation timing constraints depend on the embedded flash memory clock frequency, which directly impacts the performance. If timing constraints are too tight, the nonvolatile memory does not operate correctly, if they are too lax, the programming speed is not optimal.

The user must therefore trim the optimal programming delay through the WRHIGHFREQ parameter in the FLASH_ACCR register. Refer to *Table 25* in *Section 5.3.7: FLASH read operations* for the recommended programming delay depending on the embedded flash memory clock frequency.

The application software must check that no program/erase operation is ongoing before modifying the WRHIGHFREQ parameter.

Caution: Modifying WRHIGHFREQ while programming/erasing the flash memory might corrupt the flash memory content.

Programming errors

When a program operation fails, an error can be reported as described in *Section 5.7: FLASH error management*.

Programming interrupts

See *Section 5.8: FLASH interrupts* for details.

5.3.9 FLASH erase operations

Erase operation overview

The embedded flash memory can perform erase operations (if allowed) on 8-Kbyte user pages, or on one bank.

Note: System flash memory and read-only/OTP area cannot be erased by the application software.

The erase operation forces all nonvolatile bit cells to high state, which corresponds to the virgin state. It clears existing data and corresponding ECC, allowing a new write operation to be performed. If the application software reads back a word that has been erased or not programmed, all the bits are read at 1. No ECC error is generated if the word is located in user flash memory, while a double ECC error is raised if the word is located in read-only or OTP area.

Erase operations are similar to read or program operations except that the commands are queued in a special buffer (a one-command deep erase queue).

Erase commands are issued through the AHB configuration interface. If the embedded flash memory receives simultaneously a write and an erase request for the same bank, both operations are accepted but the write operation is executed first.

Erase and security

A user page can be erased only if it does not contain write-protected data or data currently protected by the HDP (see *Section 5.5: FLASH protection mechanisms* for details). In other words, if the application software attempts to erase a user page with at least one flash word that is protected in the bank x, the page erase operation is aborted and the WRPERRF flag is raised in the FLASH_xISR register, as described in *Section 5.7.1*.

Note: When triggering an NVSTATE change from NVCLOSE to NVOPEN an automatic erase of the whole user flash occurs, erasing HDP protected pages. See Section 5.5.3 for details.

Enabling erase operations

Before erasing a page, the application software must make sure that FLASH_xCR is unlocked. If it is not the case, an unlock sequence must be used (see *Section 5.5.1: FLASH configuration protection*).

Note: The application software must not unlock a register that is already unlocked, otherwise this register remains locked until the next system reset.

Similar constraints apply to bank erase requests.

Flash page erase sequence

To erase an 8-Kbyte user page in the bank x, proceed as follows:

1. Check and clear (optional) all the error flags due to previous programming/erase operation. Refer to *Section 5.7: FLASH error management* for details.
2. Unlock the FLASH_xCR register, as described in *Section 5.5.1: FLASH configuration protection* (only if the register is not already unlocked).
3. Set the PER bit and PSN bitfield in the corresponding FLASH_xCR register. PER indicates a page erase operation, while PSN contains the target page number in the bank x.
4. Set the START bit in the FLASH_xCR register.
5. Wait for the QW bit to be cleared in the FLASH_xSR register.

Note: If a bank erase is requested simultaneously to the page erase (BER and PER bits set), the bank erase operation supersedes the page erase operation.

Flash bank erase sequence

To erase all pages of bank x, except for those containing hide protected data, proceed as follows:

1. Check and clear (optional) all the error flags due to previous programming/erase operation. Refer to *Section 5.7: FLASH error management* for details.
2. Unlock the FLASH_xCR register, as described in *Section 5.5.1: FLASH configuration protection* (only if the register is not already unlocked).
3. Set the BER bit in the FLASH_xCR register.
4. Set the START bit in the FLASH_xCR register to start the bank erase operation. Then wait until the QW bit is cleared in FLASH_xSR register.

Note: BER and START bits can be set together, so above steps 3 and 4 can be merged.

If a page erase is requested simultaneously to the bank erase (PER bit set), the bank erase operation supersedes the page erase operation.

5.3.10 FLASH parallel operations

As the nonvolatile memory is divided into two independent banks, the embedded flash memory interface can drive different operations at the same time on each bank. For example a read, write or erase operation can be executed on bank A while another read, write or erase operation is executed on bank B.

In all cases, the sequences described in *Section 5.3.7: FLASH read operations*, *Section 5.3.8: FLASH program operations* and *Section 5.3.9: FLASH erase operations* apply.

5.3.11 Flash memory error protections

Error correction codes (ECC)

The embedded flash memory supports an error correction code (ECC) mechanism. It is based on the SECDED algorithm in order to correct single errors and detects double errors.

Per 128-bit system and user flash word the embedded flash memory uses 9 ECC bits. During each read operation from a 128-bit flash word, FLASH retrieves the 9-bit ECC information, computes the ECC of the flash word, and compares the result with the

reference value. If they do not match, the corresponding ECC error is raised as described in *Section 5.7.5: Error correction code error (SNECCERRF/DBECCERRF)*. During each program operation, a 9-bit ECC code is associated with each 128-bit data flash word, and the resulting 137-bit flash word information is written in nonvolatile memory.

For read-only and OTP areas, the embedded flash memory uses a stronger 6 ECC bits per 16-bit word. For option byte words and keys a double ECC scheme is used. For more information on read-only and OTP refer to *Section 5.3.12* and *Section 5.3.13*. For more information on option byte words and keys refer to *Section 5.4*.

Note: A double ECC error is generated for an OTP virgin word (for example a word with 22 bits at 1). When this OTP word is no more virgin, the ECC error disappears.

Cyclic redundancy codes (CRC)

The embedded flash memory implements a cyclic redundancy check (CRC) hardware module. This module checks the integrity of a given user flash memory area content.

The area processed by the CRC module can be defined either by pages or by start/end addresses. It can also be defined as the whole user flash bank.

Note: Only one CRC check operation on bank 1 or 2 can be launched at a time. To avoid corruption, do not configure the CRC calculation on the one bank, while calculating the CRC on the other bank.

When enabled, the CRC hardware module performs multiple reads by chunks of 4, 16, 64 or 256 consecutive flash-word (for example chunks of 64, 256, 1024 or 4096 bytes). These consecutive read operations are pushed by the CRC module into the required read command queue together with other AXI read requests, thus avoiding denying AXI read commands.

CRC computation uses CRC-32 (Ethernet) polynomial 0x4C11DB7:

$$X^{32}+ X^{26}+ X^{23}+ X^{22}+ X^{16}+ X^{12}+ X^{11}+ X^{10}+ X^{8}+ X^{7}+ X^{5}+ X^{4}+ X^{2}+ X + 1$$

The CRC operation is concurrent with the reading or writing of option byte user words, keys or OTP, because the same hardware is used for all those operations. To avoid corruption to the CRC computation the flash stalls above incoming requests when a CRC operation is ongoing.

The recommended sequence to configure a CRC operation in the bank x is the following:

1. Unlock FLASH_xCR register, if not already unlocked.
2. Enable the CRC feature by setting the CRC_EN bit in FLASH_xCR.
3. Program the desired data size in the CRC_BURST field of FLASH_xCRCCR.
4. Define the user flash memory area on which the CRC has to be computed. Four solutions are possible:
   - Select all user flash pages by setting the ALL_BANK bit in FLASH_xCRCCR
   - Define the area start and end addresses by programing FLASH_xCRCSADDR and FLASH_xCRCEADDR, respectively,
   - Select the targeted pages by setting the CRC_BY_PAGE bit in FLASH_xCRCCR. Also program consecutively the target page numbers using the CRC_PAGE field

of the FLASH_xCRCCR register. Set the ADD_PAGE bit after each CRC_PAGE programming.

5. When step 4 is completed start the CRC operation by setting the START_CRC bit in FLASH_xCRCCR.
6. Wait until the CRC_BUSY flag is reset in FLASH_xSR.
7. Retrieve the CRC result in FLASH_CRCDATAR.

Note: The application should avoid running a CRC on HDP user flash memory area when the hide protection level prevents access to it. Indeed, doing so may alter the expected CRC value. A special error flag defined in Section 5.7.7: CRC read error (CRCRDERRF) can be used to detect such a case.

CRC computation does not raise standard read error flags such as RDSERRF, SNECCERRF, and DBECCERRF. Only CRCRDERRF is raised.

5.3.12 FLASH one-time programmable (OTP) area

The embedded flash memory offers a 1024-byte memory area dedicated to application non-confidential one-time programmable data (OTP). It is composed of 512 words of 16 bits that cannot be erased, and can be written only once.

Note: The OTP area is virgin when the device is delivered by STMicroelectronics.

OTP data can be accessed through the AHB system port. They are organized as 16 blocks of 32 OTP words, as shown in *Table 26*. An entire OTP block can be protected (locked) from write accesses by setting the corresponding OTPL bit in the FLASH_OTPLSRP register, as shown. There is no special read protection mechanism on the OTP area.

*Note: The OTP block locking operation is **irreversible** and independent from the life-cycle management described in Section 5.5.3.*

A block can be write-protected whether or not it has been programmed (even partially).

Table 26. Flash memory OTP organization

OTP block	AHB address	AHB word		Lock bit
		[31:16]	[15:0]	
Block 0	0x08FF F000	OTP001	OTP000	OTPL[0]
	0x08FF F004	OTP003	OTP002	
	...			
	0x08FF F03C	OTP031	OTP030	
Block 1	0x08FF F040	OTP033	OTP032	OTPL[1]
	0x08FF F044	OTP035	OTP034	
	...			
	0x08FF F07C	OTP063	OTP062	
Block 2	0x08FF F080	OTP065	OTP064	OTPL[2]
	0x08FF F084	OTP067	OTP066	
	...			
	0x08FF F0BC	OTP127	OTP126	

Table 26. Flash memory OTP organization (continued)

OTP block	AHB address	AHB word		Lock bit
		[31:16]	[15:0]	
...				
Block 15	0x08FF F3C0	OTP481	OTP480	OTPL[15]
	0x08FF F3C4	OTP483	OTP482	
	...			
	0x08FF F3FC	OTP511	OTP510	

OTP error protection

OTP data are organized as 16 blocks of 32 OTP words, as shown on *Table 26*. Each 16-bit half-word is protected by 6 bits of ECC. Hence the application must avoid overwriting a 16-bit half-word that has already been programmed, as it is leading to systematic ECC error. Similarly, do not write twice a 16-bit OTP word, as it could lead to systematic ECC error.

When reading OTP data with a single error corrected or a double error detected, the embedded flash memory reports the corresponding error, as described in *Section 5.7.5: Error correction code error (SNECCERRF/DBECCERRF)*.

When reading OTP data that has not been written by the application software (that is, virgin OTP), the ECC correction reports a double error detection (DBECCERRF), and all 1's are returned. It is therefore recommended that the application always writes the OTP data before trying to read it.

OTP write sequence

Follow the sequence below to write an OTP word:

1. Check the protection status of the target OTP word (see *Table 26*). The corresponding OTPLi bit must be cleared in FLASH_OTPLSR register.
2. Unlock FLASH_xCR if the register is not already unlocked, and then set the PG_OTP bit in the FLASH_xCR register.
3. Write two OTP words (32 bits) corresponding to the 4-byte aligned address shown in *Table 26*. Alternatively, the application software can program separately the 16-bit MSB or 16-bit LSB. In this case the first 16-bit write operation starts immediately without waiting for the second one.
4. Check that the QW bit in FLASH_xSR has been set and wait until it is cleared.
5. Read back the target OTP word to confirm the value
6. Optionally, lock the OTP block using OTPLi bit in FLASH_OTPLSRP in order to prevent further data changes. PG_OTP bit in the FLASH_xCR register can also be cleared if needed.

Note: To avoid data corruption, it is important to complete the OTP write process (for example by reading back the OTP value), before starting a new option byte change.

OTP write operations can trigger PGSERRF or WRPERRF errors (see Section 5.7).

Writing OTP data at byte level is not supported and generates a bus error.

5.3.13 FLASH read-only (RO) area

The embedded flash memory offers a 512-byte memory area dedicated to non-confidential and read-only information usable by application software.

Read-only area can be accessed through the AHB system port. It is organized as shown in *Table 27*. This information is provisioned by STMicroelectronics during the device manufacturing.

Table 27. read-only public data organization

AHB address (read-only)	Read-only data name and usage
0x08FF F800	U_ID[31:0]
0x08FF F804	U_ID[63:32]
0x08FF F808	U_ID[95:64]
0x08FF F80C	Flash memory size[15:0] || Package code[15:0]
0x08FF F810 to 0x08FF F9FF	Reserved information

Read-only area error protection

Read-only area is protected by a robust ECC scheme. When reading a 32-bit word in this area with a single error corrected or a double error detected, the embedded flash memory reports the corresponding error, as described in *Section 5.7.5: Error correction code error (SNECCERRF/DBECCERRF)*.

5.3.14 FLASH bank and register swapping

Allocation of physical banks 1 & 2 to embedded flash user banks A & B depends on the SWAP_BANK bit in FLASH_xSR registers (see *Section 5.9.5* for details). Application can change this allocation by writing to the SWAP_BANK option in FLASH_OBW1SRP register, for example after a firmware upgrade to restart the device on a new firmware.

Since some registers of the embedded flash are not specific to any particular physical bank they are not affected by the SWAP_BANK bit, as shown on *Table 28*. *Table 29* summarizes the impact of SWAP_BANK bit on embedded flash's typical operation flags.

Table 28. Flash registers versus swapping option

Register name (x= A or B)	Affected by SWAP_BANK(1)
FLASH_xKEYR	Yes
FLASH_xCR, FLASH_xSR	Yes
FLASH_xIER/xISR/xICR	Yes
FLASH_xCRCCR	Yes
FLASH_xCRCSADDR	
FLASH_xCRCEADDR	
FLASH_xECCSFADDR	Yes
FLASH_xECCDFADDR	

Table 28. Flash registers versus swapping option

Register name (x= A or B)	Affected by SWAP_BANK(1)
FLASH_xWRPSR/xWRPSRP	Yes(2)
FLASH_xHDPSR/xHDPSRP	
FLASH_xHDPEXTR	Yes
FLASH_OPTKEYR	No
FLASH_OPTCR	No
FLASH_OPTISR/OPTICR	
FLASH_OTPLSR/OTPLSRP	No
FLASH_NVSR/NVSRP	
FLASH_ROTSR/ROTSRP	
FLASH_OBKCR/OBKCLR	No
FLASH_OBKDR0 to OBKDR3	
FLASH_OBW1SR/OBW1SRP	
FLASH_ACCR	

1. No means: behavior of the register is unchanged whether SWAP_BANK = 0 or 1.
2. This protection is tied to the underlying physical bank. In case of swapping the register values is also swapped to follow the physical bank.

Table 29. Flash operation flags versus swapping option

FLASH operation	SWAP_BANK = 0		SWAP_BANK = 1		Comment
	xSR	xISR	xSR	xISR	
User flash R/W/E (bank 1)	A		B		Typical flags: QW, IS_PROGRAM, IS_ERASE, error flags
User flash R/W/E (bank 2)	B		A		
System flash R/W (bank 1)	-	A	-	B	Only error flags are impacted as no write or erase operations are allowed.
System flash R/W (bank 2)	-	B	-	A	
Option Byte change	A		B		Typical flags: QW, EOPF, IS_OPTCHANGE, OPTERRF, OBLERRF
Option byte key change	B	-	A	-	Typical flags: QW
OTP R/W	A		B		Typical flags: QW, PG_OTP, WRPERRF

SWAP_BANK bit usage

The read-only SWAP_BANK bit in FLASH_xSR registers are loaded from the SWAP_BANK option bit only after system reset or power on reset. To change it, for example to apply a new firmware update, respect the sequence below:

1. Unlock FLASH_OPTCR register as described in *Section 5.5.1: FLASH configuration protection* (only if the register is not already unlocked).
2. Set the new desired SWAP_BANK value in the FLASH_OBW1SRP register.
3. Set the OPTSTART bit in the FLASH_OPTCR register.
4. Wait for the QW bit to be cleared in the FLASH_xSR register. Once cleared, the corresponding FLASH_OBW1SR registers are updated, if security allows it. The SWAP_BANK bit in FLASH_xSR registers has not yet been modified however, making the bank swapping not yet effective.
5. Force a system reset or a power on reset. Following this reset the embedded flash updates SWAP_BANK in FLASH_xSR according to the FLASH_OBW1SR value. The bank swapping is now effective and the new firmware executes.

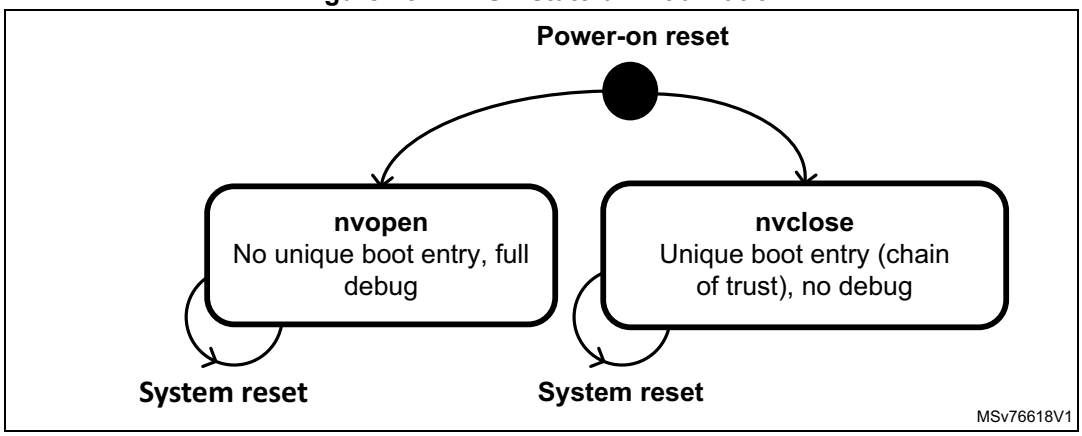
Note: The SWAP_BANK option bit in FLASH_OBW1SRP can be modified whatever the product state. ST RSS can however lock its value using the BOOT_LOCK option in FLASH_ROTSR.

5.3.15 FLASH reset and clocks

Reset management

Following a power-on reset, the embedded flash selects two security states, as shown on *Figure 15*. For more details on those states, refer to *Section 5.5.3*.

Figure 15. FLASH stateful initialization



Embedded flash memory stateful initialization is based on the overloading of option bytes default values by nonvolatile values automatically loaded when the flash_rst signal rises (see *Table 22*). During this loading sequence, the other parts of the device remain under reset and the embedded flash memory is not accessible from its interfaces.

The embedded flash memory can be reset at any time by the application through the RCC peripheral, with the following effects:

- All registers, except for user option byte registers, are cleared, including read and write latencies. User option byte words are listed in *Section 5.4.5*.
- Most control registers are automatically protected against write operations, as described in *Section 5.5.1: FLASH configuration protection*.

It is important to note that the contents of the flash memory (except option bytes) are not guaranteed if a power-on-reset occurs during a flash memory write or erase operation. For option bytes a valid content is always guaranteed, as embedded flash uses the old option

byte values when an option byte modification is interrupted by a reset. In this case a new option byte change request must be issued to modify those option bytes.

Clock management

The embedded flash memory uses the AXI interface clock. Depending on the device clock and internal supply voltage, specific read and write latency settings usually need to be set in the flash access control register (FLASH_ACCR), as explained in *Section 5.3.7: FLASH read operations* and *Section 5.3.8: FLASH program operations*.

5.3.16 Tamper management

When option bytes stored in either FLASH_NVSR or FLASH_ROTSR are corrupted an internal tamper 15 is issued to the TAMP peripheral.

When a tamper event is activated by the TAMP peripheral all option byte keys stored in the embedded flash read as zero. It is also not possible to modify or delete option byte keys. Those protections are de-activated when the tamper event is cleared in TAMP.

5.4 FLASH option bytes

5.4.1 About option bytes

The embedded flash memory includes a set of nonvolatile option bytes that are either freely modified through configuration registers (option byte words), or are managed as nonvolatile keys with a special temporal isolation protection mechanism.

This section explains:

- How option byte words and keys are loaded
- How application software can change option byte words, under which conditions
- How application software can program option byte keys
- What is the detailed list of option byte words, together with their default factory values (that is, before the first option byte change).

5.4.2 Option byte loading

When the device is first powered, the embedded flash memory automatically loads all the option byte words. During the option byte loading sequence, the device remains under reset and the embedded flash memory cannot be accessed. In case of security issue, the option bytes may be loaded with default values (see *Section 5.4.3: Option byte words modification*).

When an ECC issue is detected during this option byte loading sequence, the OBLERRF flag is raised as described in *Section 5.7.9*.

5.4.3 Option byte words modification

Changing option byte words

An option byte word change operation can modify the configuration and the protection settings saved in the nonvolatile option byte area of memory bank A, if allowed.

The embedded flash memory features two sets of option byte words registers:

- The first register set contains the current values of the option bytes. Their names have the SR extension. All “SR” registers are read only. Unless documented otherwise, their values are automatically loaded from the nonvolatile memory after power-on reset, wake-up from system standby or after a successful option byte change operation.
- The second register set allows the modification of the option bytes. Their names end with the SRP extension. All “SRP” registers can be accessed in read/write mode.

Note: When the OPTLOCK bit in the FLASH_OPTCR register is set, writes to SRP registers are ignored.

When OPTSTART bit is set the embedded flash memory checks if at least one option byte needs to be programmed by comparing the current values (in registers “SR”) with the new ones (in corresponding registers “SRP”). If a change is detected, if associated change conditions are met (see *Changing security option bytes*), the embedded flash memory launches the option byte modification in its nonvolatile memory and update the “SR” register.

If one of the conditions described in *Changing security option bytes* is not respected, the embedded flash memory sets the OPTERRF flag in the FLASH_OPTISR register and aborts the option byte change operation. In this case, the “SRP” registers are not overwritten by the current option value in the corresponding “SR” registers. The user application can check what was wrong in their configuration.

Unlocking the option byte modifications

After reset, the OPTLOCK bit is set and the FLASH_OPTCR is locked. As a result, the application software must unlock the option configuration register before attempting to change the option byte words, setting OPTSTART bit. The FLASH_OPTCR unlock sequence is described in *Section 5.5.1*

Option byte modification sequence

To modify an option byte word, follow the sequence below:

1. Unlock FLASH_OPTCR register as described in *Section 5.5.1: FLASH configuration protection* (only if the register is not already unlocked).
2. Write the desired new option byte values in the corresponding option byte word registers (FLASH_xxSRP).
3. Set the OPTSTART bit in the FLASH_OPTCR register.
4. Wait until QW is cleared in FLASH_xSR register. Once cleared, the corresponding FLASH_xxSR registers are updated, if security allows it (see next section).

If a reset or a power-down occurs while the option byte words modification is ongoing, the original option byte value are kept. A new option byte modification sequence is required to program the new value.

The application must not modify the SRP registers until the option byte change operation has been completed.

Changing security option bytes

On top of OPTLOCK bit, there is a second level of protection for security-sensitive option bytes. They are described hereafter.

- **Page write protection (WRPG)**

  Those user options managing groups of page write protection in bank x can be changed using FLASH_xWRPSRP register, if the application is RSS or the immutable root of trust.

- **Nonvolatile state (NVSTATE)**

  A detailed description of NVSTATE option bits is given in *Section 5.5.3*. The following rules must be respected to modify NVSTATE using the FLASH_NVSRP register.

  – Writes to the FLASH_NVSRP register are ignored if HDPL is different from 0 or 1 in the SBS_HDPLSR register.

  – NVSTATE option change is triggered only after the RAM erase signal has confirmed that Backup RAM has been successfully erased.

  – When NVSTATE = NVOPEN, option byte change to NVCLOSE is allowed. Writing any other value than 0x51 triggers an OPTERRF flag, without option change.

  – When NVSTATE = NVCLOSE, option byte change to NVOPEN is allowed, triggering a global erase explained in *Section 5.5.3*. Writing any other value than 0xB4 triggers an OPTERRF flag, without option change.

- **OEM provisioned (OEM_PROVD)**

  As defined in *Table 34: Boot level and HDP area protections,* OEM_PROVD defines the protection of the HDP area. OEM_PROVD can only be changed by RSS in FLASH_ROTSRP.

- **Hide protection area size (HDP_AREA_START and HDP_AREA_END)**

  These user options configure the size of the user flash area in bank x that can be accessed while HDPL equals to the value defined in *Section 5.5.4*. Writes to FLASH_xHDPSRP register are ignored if HDPL is different from 0 or 1 in the SBS_HDPLSR register.

- **Debug authentication method (DBG_AUTH)**
  DBG_AUTH defines if the device is closed or locked. See *Section 5.5.3* for details. It has no hardware effect in embedded flash memory.
  DBG_AUTH can only be changed by RSS in FLASH_ROTSRP.

- **Bank swapping (SWAP_BANK)**
  BANK_SWAP defines if the device needs to swap its physical banks. See *Section 5.3.14* for details. Option byte change is ignored when BOOT_LOCK is set in FLASH_ROTSR.

Note: For all user option bytes in this section: default values are loaded when a double ECC error occurs. OBLERRF error flag is also raised.

Hide protection area end address in system flash is fixed by STMicroelectronics.

5.4.4 Option byte key management

The device can manage 32 key slots of 16-bytes per HDP level (up to level 2, included).

Unlocking the option byte key modifications

After reset, the OPTLOCK bit is set and the FLASH_OBKCR is locked. As a result, the application software must unlock the option byte key configuration register before

attempting to change the option byte keys, setting KEYSTART bit. The FLASH_OPTCR unlock sequence is described in *Section 5.5.1*

Option byte key modification

To write a virgin option byte key, follow the sequence below:

1. Unlock the FLASH_OPTCR register as described in *Section 5.5.1: FLASH configuration protection* (only if the register is not already unlocked).
2. Write the desired option byte key value in FLASH_OBKDRx registers.
   - Key[31:0] goes to OBKDR0[31:0], Key[63:32] to OBKDR1[31:0], Key[95:64] to OBKDR2[31:0], and Key[127:96] to OBKDR3[31:0].
3. Fill the following information to FLASH_OBKCR register:
   - OBKINDEX[4:0]: index of the new key, corresponding to the selected HDP level.
   - NEXTKL[1:0]: when HDPL = 1 in SBS_HDPLSR register the RoT application can set those bit to 0x1 to provision a key for the next HDPL level (2).
   - Set both KEYSTART and KEYPROG bits to start the programming sequence.
4. Wait until the KEYSTART bit is cleared in FLASH_OBKCR register. Once cleared read back the key to verify its value (recommended). See *Option byte key loading*.
   - In SBS_HDPLSR, if HDPL = 3 during above step 3 the key returns all ones (programming was ignored). No error is triggered.

Note: When HDPL value changes in SBS peripheral FLASH_OBKDRx registers are cleared.

For a given hide protection level, an authorized application cannot overwrite an option byte key that was successfully written before. Therefore before modifying this key it must be erased, using the sequence below.

1. Unlock the FLASH_OPTCR register as described in *Section 5.5.1: FLASH configuration protection* (only if the register is not already unlocked).
2. In FLASH_OBKCLR register set the key erase bit on each existing key that the application wants to overwrite/ update.
3. In FLASH_OBKCR register, set both KEYSTART and KEYERASE bits to start the erasing of all selected keys.
4. Wait until the KEYSTART bit is cleared in the FLASH_OBKCR register. Once cleared the FLASH_OBKCLR register is zeroed.

If the application tries to write a key that is not virgin or erased OBKERRF error flag is set (see *Section 5.7.10*).

Option byte key loading

The following sequence is required to read a 16-byte option byte key stored in embedded flash memory:

1. Fill the following information to FLASH_OBKCR:
   - OBKINDEX[4:0]: index of the key for this hide protection level.
   - NEXTKL[1:0]: when HDPL = 1 in SBS_HDPLSR register, the RoT application can set those bits to 0x1 to read a key provisioned for the next HDPL level (2).
   - Set KEYSTART bit with KEYPROG bit cleared to start the read sequence.
2. When KEYSTART is cleared read the value of the key using the FLASH_OBKDRx registers. A virgin key returns all ones.
   - In SBS_HDPLSR, if HDPL = 3 during above step 1 the key returns all ones (virgin key). No error is triggered.

If OBKERRF error flag is set following a key loading FLASH_OBKDRx registers reads as zero. After clearing the error try loading the option byte key again.

Note: When HDPL value changes in SBS peripheral FLASH_OBKDRx registers are cleared.

5.4.5 Option byte user words overview

Table 30 lists all the user option bytes managed through the embedded flash memory registers, as well as their default values before the first option byte change (default factory value). They are configured by the end user depending on the application requirements. Some option bytes might have been initialized by STMicroelectronics during the manufacturing stage.

Table 30. Option byte user words organization

Register	Bitfield															
FLASH_OBW1SR[31:16]	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	SWAP_BANK	Res.	VDDIO_HSLV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG_FZ_STOP	IWDG_FZ_STBY	Res.
Default factory value	*0*	*0*	*1*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*1*	*0*
FLASH_OBW1SR[15:0]	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Res.	Res.	Res.	Res.	Res.	ECC_ON_SRAM	Res.	Res.	NRST_STBY	NRST_STOP	Res.	IWDG_HW	BOR_LEV		Res.	Res.
Default factory value	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*1*	*0*	*1*	*0*	*0*	*0*	*0*
FLASH_AHDPSR[31:16]	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	Res.	Res.	Res.	Res.	HDP_AREA_END											

Table 30. Option byte user words organization (continued)

Register	Bitfield															
Default factory value	*0*	*0*	*0*	*0*	*0x000*											
FLASH_AHDPSR[15:0]	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Res.	Res.	Res.	Res.	HDP_AREA_START											
Default factory value	*0*	*0*	*0*	*0*	*0x0FF*											
FLASH_BHDPSR[31:16]	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
-	Res.	Res.	Res.	Res.	HDP_AREA_END											
Default factory value	*0*	*0*	*0*	*0*	*0x000*											
FLASH_BHDPSR[15:0]	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
-	Res.	Res.	Res.	Res.	HDP_AREA_START											
Default factory value	*0*	*0*	*0*	*0*	*0x0FF*											
FLASH_AWRPSR[31:16]	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
(STM32H7PxxI devices only)	WRPG[127:124]	WRPG[123:120]	...										WRPG[71:68]	WRPG[67:64]	WRPG[71:68]	WRPG[67:64]
Default factory value	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
FLASH_AWRPSR[15:0]	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
-	WRPG[63:60]	WRPG[59:56]	...										WRPG[15:12]	WRPG[11:8]	WRPG[7:4]	WRPG[3:0]
Default factory value	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
FLASH_BWRPSR[31:16]	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
(STM32H7PxxI devices only)	WRPG[127:124]	WRPG[123:120]	...										WRPG[71:68]	WRPG[67:64]	WRPG[71:68]	WRPG[67:64]
Default factory value	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
FLASH_BWRPSR[15:0]	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

Table 30. Option byte user words organization (continued)

Register	Bitfield															
-	WRPG[63:60]	WRPG[59:56]	...										WRPG[15:12]	WRPG[11:8]	WRPG[7:4]	WRPG[3:0]
Default factory value	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
FLASH_OTPLSR[15:0]	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	OTPL[511:481]	OTPL[480:449]	...												OTPL[63:32]	OTPL[31:0]
Default factory value	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
FLASH_NVSR[15:0]	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NVSTATE[7:0]							
Default factory value	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*0*	*1*	*1*	*0*	*1*	*0*	*0*
FLASH_ROTSR[31:16]	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
-	BOOT_LOCK[7:0]								...							
Default factory value	*1*	*0*	*1*	*1*	*0*	*1*	*0*	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
FLASH_ROTSR[15:0]	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
-	DBG_AUTH[7:0]								OEM_PROVD[7:0]							
Default factory value	*1*	*0*	*1*	*1*	*0*	*1*	*0*	*0*	*0*	*1*	*1*	*0*	*1*	*0*	*1*	*0*

5.4.6 Description of user option byte word

The general-purpose option bytes that can be used by the application are listed below. They are accessed through the FLASH_OBWSR register.

- Watchdog management
  - IWDG_FZ_STOP: independent watchdog IWDG counter active in Stop mode if 1 (stop counting or freeze if 0)
  - IWDG_FZ_STBY: independent watchdog IWDG counter active in Standby mode if 1 (stop counting or freeze if 0)
  - IWDG_HW: hardware (0) or software (1) IWDG watchdog control selection

Note: If the hardware watchdog "control selection" feature is enabled (set to 0), the watchdog is automatically enabled at power-on, thus generating a reset unless the watchdog key register is written to or the down-counter is reloaded before the end-of-count is reached.

Depending on the configuration of IWDG_STOP and IWDG_STBY options, the IWDG can continue counting (1) or not (0) when the device is in Stop or Standby mode respectively.

When the IWDG is kept running during Stop or Standby mode, it can wake up the device from these modes.

- Reset management
  - BOR_LEV: Brownout level option, indicating the supply level threshold that activates/releases the reset
  - NRST_STBY: The independent watchdog generates a reset when entering Standby mode if cleared to 0
  - NRST_STOP: The independent watchdog generates a reset when entering Stop mode if cleared to 0.

Note: Whenever a Standby (respectively Stop) mode entry sequence is successfully executed, the device is reset instead of entering Standby (respectively Stop) mode if NRST_STBY (respectively NRST_STOP) is cleared to 0.

- Bank swapping option (SWAP_BANK). See *Section 5.3.14* for details.
- Device options
  - VDDIO_HSLV: enables the configuration of pads below 2.5 V for VDDIO power rail if set to 1.
  - VDDMMC_HSLV: enables the configuration of pads below 2.7 V for VDDMMC power rail if set to 1.

When STMicroelectronics delivers the device, the values programmed in the general-purpose option bytes are the following:

- Watchdog management
  - IWDG reset active in Standby and Stop modes (option value = 0x1)
  - IWDG not automatically enabled at power-on (option byte value = 0x1)
- Reset management:
  - BOR: brownout level option (reset level) equals 2.1 V (option byte value = 0x0). A reset is not generated when the device enters Standby or Stop low-power mode (option byte value = 0x1)
- The speed of the device I/Os is optimized when the device voltage is low (VDDIO_HSLV = VDDMMC_HSLV = 1)

Refer to *Section 5.9: FLASH registers* for details.

5.4.7 Description of security option bytes

The option bytes that are used by root of trust application to enhance security are listed below:

- NVSTATE[7:0]: Nonvolatile state (see *Section 5.5.3* for details).
- WRPG[127:0]: Write protection option of user flash pages (one word per bank). It is active low. Refer to *Section 5.9.26* for details.
- OTPL[15:0]: Write protection option of OTP blocks. It is active high. Refer to *Section 5.3.12* for details.
- HDP_AREA: Hide protection user flash area definitions. Refer to *Section 5.5.4*.
  - HDP_AREA_START (respectively HDP_AREA_END) contains the first (respectively last) 256-byte block of the HDP area.

When STMicroelectronics delivers the device, the values programmed in the security option bytes are the following:

- NVOPEN device (NVSTATE value = 0xB4)
- SWAP_BANK = 0
- HDP area protection disabled (start addresses higher than end addresses)
- Write protection disabled (all option byte bits set to 1)
- All OTP blocks are writable, as they are virgin (all option byte bits set to 0)

Refer to *Section 5.9: FLASH registers* for details.

5.5 FLASH protection mechanisms

Since sensitive information is stored in the flash memory, it is important to protect it against unwanted operations such as reading confidential areas, illegal programming of immutable pages, or malicious flash memory erasing.

For this purpose FLASH implements the following protection mechanisms:

- Configuration protection
- User flash write protection
- Device nonvolatile security life cycle and application temporal isolation management
- OTP locking

This section provides a detailed description of all these security mechanisms.

5.5.1 FLASH configuration protection

The embedded flash memory uses hardware mechanisms to protect the following assets against unwanted or spurious modifications (e.g. software bugs):

- Option bytes change
- Write operations
- Erase commands
- Interrupt masking

More specifically, write operations to embedded flash memory control registers (FLASH_xCR, FLASH_OPTCR and FLASH_OBKCR) are not allowed after reset.

The following sequence must be used to unlock FLASH_xCR register:

1. Program KEY1 to 0x45670123 in the FLASH_xKEYR key register.
2. Program KEY2 to 0xCDEF89AB in FLASH_xKEYR key register.
3. LOCK bit is now cleared and FLASH_xCR is unlocked.

The following sequence must be used to unlock FLASH_OPTCR and FLASH_OBKCR registers. Registers unlocked with FLASH_OPTCR are listed in *Table 31*.

1. Program OPTKEY1 to 0x08192A3B in FLASH_OPTKEYR option key register.
2. Program OPTKEY2 to 0x4C5D6E7F in FLASH_OPTKEYR option key register.
3. OPTLOCK bit is now cleared. FLASH_OPTCR and FLASH_OBKCR registers are unlocked.

Any wrong sequence locks up the corresponding register/bit until the next system reset, and generates a bus error.

The above registers can be locked again by software by setting the LOCK bit in the corresponding control register.

The FLASH_xCR register remains locked and a bus error is generated when the following operations are executed:

- Programming a third key value
- Writing to a different register belonging to the same bank than FLASH_xKEYR before FLASH_xCR has been completely unlocked (KEY1 programmed but KEY2 not yet programmed)
- Writing less than 32 bits to KEY1 or KEY2.

Similarly the FLASH_OPTCR, FLASH_OBKCR, and related registers (see *Table 31*) remain locked and a bus error is generated when the following operations are executed:

- Programming a third key value
- Writing to a different register before FLASH_OPTCR or FLASH_OBKCR have been completely unlocked (OPTKEY1 programmed but OPTKEY2 not yet programmed)
- Writing less than 32 bits to OPTKEY1 or OPTKEY2.

The embedded flash memory configuration registers protection is summarized in *Table 31*. Registers not present in this table are not protected by any key.

Table 31. Flash interface register protection summary

Register name	Unlocking register	HDP level(1)	Protected asset
FLASH_xCR	FLASH_xKEYR	Any	Flash write operations, erase commands
FLASH_xIER	FLASH_xCR		Interrupt generation masking sources
FLASH_xCRCCR			Flash bank CRC operations
FLASH_OPTCR	FLASH_OPTKEYR		Option bytes changes
FLASH_OBKCR			Option byte key changes
FLASH_NVSRP	FLASH_OPTCR	0 or 1	Nonvolatile state change
FLASH_ROTSRP		0	Root of trust parameter change
FLASH_xHDPSRP		0 or 1	Hide protect area change for bank x
FLASH_OBWSRyP		Any	User option byte word y change
FLASH_xWRPSRP			User flash write protection change for bank x
FLASH_OTPLSRP	FLASH_OPTCR(2)	Any	OTP write protection

1. Required HDP level in the device
2. Once set, the OTP Block Lock bits cannot be reset by the application.

5.5.2 Write protection

The purpose of embedded flash memory write protection is to prevent unwanted modifications to embedded nonvolatile code and/or data.

Any group of four consecutive 8 Kbyte pages can be independently write-protected or unprotected by clearing/setting the corresponding WRPG bit in the FLASH_xWRPSR register.

A write-protected group of pages can neither be erased nor programmed. As a result, a full bank erase cannot be performed if one group of pages is write-protected, unless an NVSTATE transition to NVOPEN is triggered.

The embedded flash memory write-protection user option bits can be modified when HDPL=0 or 1 in SBS_HDPLSR register, using FLASH_xWRPSRP.

Note: The HDP area in user flash is write and erase protected, while system flash cannot be erased by user application.

Write protection flags remain valid for the physical bank, so they follow the bank when it is swapped. See *Section 5.3.14* for details.

Write protection errors are documented in *Section 5.7: FLASH error management*.

5.5.3 Life cycle management

Introduction

Figure 16 shows how embedded flash manages the nonvolatile security life cycle of the device.

Figure 16. Life cycle management scheme

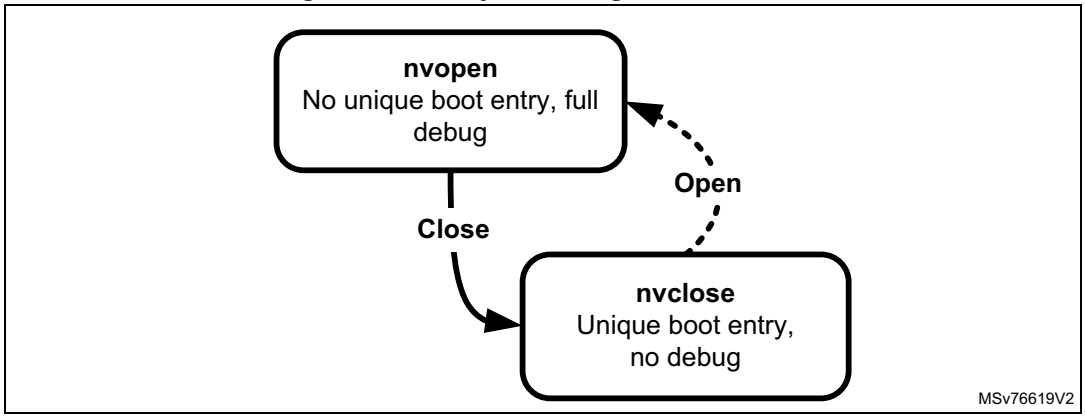


Table 32 details the nonvolatile security lifecycle NVSTATE and its effects on the product.

Table 32. Flash security lifecycle definition

NVSTATE	Boot address(1)	Debug level	Regression to
NVOPEN	0x0800 0000 or Bootloader	Unconstrained	n/a
NVCLOSE	0x1FF0 0000	Closed (2)	NVOPEN

1. Enforced outside flash.
2. Application debug is closed, with option described in *Table 33*. STMicroelectronics engineering test modes are not available.

Table 33 details the other nonvolatile security option bytes and their effects on the product.

Table 33. Option bytes versus lifecycle definition

Product state	NVSTATE	OEM_PROVD	DBG_AUTH	Description
Open	NVOPEN (0xB4)	Ignored (0x6A)	Ignored (0x6F)	Device delivered by ST. User flash memory usage is open, and debug access is enabled (see *Table 32*).
Provisioning	NVCLOSE (0x51)	Not provisioned (0x6A)	Closed (0x6F)	Device on the production line. RSS user function *DataProvisioning* is used to provision regression password, and optionally root of trust OBKeys.
Locked		Provisioned (0xB4)	Locked (0xB4)	Final product with no debug and no regression possible (which means transition to another product state is forbidden).
Closed			Closed (0x6F or 0x8A)	Final product with no debug out of reset, and transition to open state is possible (see *Flash opening*).

Flash opening

When the regression process code on the device (system flash code, or OEM code running in HDPL1) receives a valid regression request, it must first erase OBKeys (except the regression password) using the method defined in *Option byte key modification*. After confirming the keys have been destroyed, the regression process code opens the device by writing 0xB4 to FLASH_NVSRP, then triggers an option byte change as defined in *Section 5.4.3*.

Detecting this option byte changes the flash memory requests:

- The Backup RAM to be zeroed,
- An automatic erase of the whole user flash (write protected or not),
- A reset to their default manufacturing values (protections disabled) for the write protection registers (FLASH_xWRPSR), and the hide protection registers (FLASH_xHDPSR, FLASH_xHDPEXTR).

Once the embedded flash memory changes the FLASH_NVSRP the device triggers a power-on-reset, as described in *Figure 15*.

The system flash memory and the read-only region are not affected by flash opening and remain unchanged. If the regression process code is an OEM code running in HDPL1, no option byte stored in FLASH_ROTSR can be changed.

If the regression request fails (NVSTATE = NVCLOSE unchanged), the application must trigger it again using the same password.

Flash closing

Writing 0x51 to the FLASH_NVSRP register triggers a change of NVSTATE from NVOPEN to NVCLOSE, confirmed when NVSTATE=0x51 in the FLASH_NVSR register.

5.5.4 Hide protection (HDP)

Hide protection in system flash

System flash pages included between 0x1FF0 0000 and SEC_AREA_END addresses store STM32 secure bootloader and password management code.

Embedded flash system area can only be accessed by Cortex M7 in read or execute. The hide protected area can only be accessed when NVSTATE is NVCLOSE and when hide protection level in SBS_HDPLSR is 0.

In all other cases, one of the following events is triggered:

- Data read transactions return zero. Data write transactions are ignored. No bus error is generated but an error flag is raised (RDSERR for read, WRPERR for write).
- Read instruction transactions generate a bus error and the RDSERR error flag is raised.

Embedded flash system area is immutable, that is, it cannot be erased by any application code. This area includes the system bootloader area from SEC_AREA_END to 0x1FF1FFF.

Note: When FLASH transitions from NVCLOSE to NVOPEN state, the embedded flash system area is kept.

Hide protection in user flash

User flash pages included between HDP_AREA_START and HDP_AREA_END addresses are used to store the root of trust code and data, with the associated keys stored in HDPL1 option byte keys.

User flash pages located inside an active HDP extension area are used to store updatable root of trust code and data, with the associated keys stored in HDPL2 option byte keys.

Embedded flash hide protected (HDP) user area and active HDP extension area can only be accessed:

- By Cortex M7 in read, write or execute
- According to the conditions defined in *Table 34*

Table 34. Boot level and HDP area protections

<table>
<tr><th rowspan="2">Product state</th><th colspan="2">FLASH user options(1)</th><th colspan="3">HDP level(2) to activate protection</th><th rowspan="2">Boot address (HDPL)</th><th rowspan="2">Comment</th></tr>
<tr><th>NV STATE</th><th>OEM_ PROVD</th><th>HDP area in system flash</th><th>HDP area in user flash</th><th>HDP extension area</th></tr>
<tr><td>Open</td><td>NV OPEN</td><td rowspan="2">not 0xB4</td><td rowspan="3">>0</td><td>>1</td><td rowspan="3">>2 (if active)</td><td>User flash or bootloader (HDPL1)</td><td>RSS not visible</td></tr>
<tr><td>Provisioning</td><td rowspan="2">NV CLOSE</td><td>>0(3)</td><td rowspan="2">RSS (HDPL0)</td><td>Via RSSLib/ DataProvisioning service user provisions OBKeys and firmware in user flash.</td></tr>
<tr><td>Closed or Locked</td><td>0xB4</td><td>>1</td><td>Firmware provisioned in user flash. All related OBKeys are also provisioned.</td></tr>
</table>

1. In FLASH_NVSR and FLASH_ROTSR registers
2. Level for which hide protection is activated. Current HDP level is available in SBS_HDPLSR register.
3. Because of this condition firmware in user flash cannot execute in this product state (OBKeys not yet provisioned).

For more details on HDPL level following a reset, refer to the *SBS* section of the manual.

HDP programming in user flash

HDP user flash area of bank x (x = A or B) is programmed in FLASH_xHDPSRP by setting the HDP_AREA_START and HDP_AREA_END so that the END address is strictly higher than the START address.

HDP_AREA_START and HDP_AREA_END are defined with a granularity of 256 bytes. This means that the actual HDP user flash area size (in bytes) is defined by:

(HDP_AREA_END - HDP_AREA_START + 1) x 256

As an example, to set the HDP user flash area on the first 8 Kbytes (that is, from address 0x0800 0000 to address 0x0800 1FFF, both included), the embedded flash memory must be configured as follows:

HDP_AREA_START[15:0] = 0x0

HDP_AREA_END[15:0] = 0x001F

The HDP user flash area size defined above is equal to:

(HDP_AREA_END - HDP_AREA_START + 1) x 256 = 32 x 256 bytes = 8 Kbytes.

The minimum HDP user flash area that can be set is 32 flash words (or 512 bytes). The maximum area is the whole user flash memory bank, configured by setting HDP_AREA_START = HDP_AREA_END.

Note: It is recommended to align the HDP user area size with 8 Kbytes flash page granularity to always be able to erase non-HDP code and data.

It is possible to disable HDP protections by setting HDP_AREA_END lower than HDP_AREA_START.

HDP properties in user flash

- Embedded flash hide protected user area can only be accessed in the conditions described in *Table 34*. In all other cases, one of the following events is triggered:
  - Data read transactions return zero. Data write transactions are ignored. No bus error is generated but an error flag is raised (RDSERR for read, WRPERR for write).
  - Read instruction transactions generate a bus error and the RDSERR error flag is raised.
- HDP user flash area is erase-protected. As a result:
  - No erase operations to a page located in this area are possible (including the page containing the area end address), unless HDPL=0 or 1.
  - In HDPL = 2 or 3 the only possibility to fully erase the user flash bank is to perform a device opening sequence (see *Section 5.5.3*).
- Rules to change the HDP user flash area are detailed in *Changing option byte words* in *Section 5.4.3*.
- When FLASH transitions from NVCLOSE to NVOPEN state, the whole user flash (hide protected or not) is automatically erased, with all the HDP area disabled (see *Section 5.5.3*).
- HDP user flash area remains valid for the physical bank, so it follows the bank when it is swapped. See *Section 5.3.14* for details.

For more information on HDP user flash area errors, refer to *Section 5.7: FLASH error management*.

HDP extension

For each bank x, the nonvolatile HDP area can be extended, using a volatile extension defined in FLASH_xHDPEXTR register (see *Section 5.9.19*). Accesses to an active HDP extension area are ignored (for writes) or zeroed (for reads) when HDPL is different from 0, 1 or 2 in the SBS_HDPLSR register.

Like HDP user flash area an active HDP area extension is erase-protected. The same rules as HDP user flash area apply when HDPEXT_ACCDIS[7:0] is different from 0xA3 (default), and HDP_AREA_EXT[11:0] is different from zero (default).

Controlling the size of the HDP extended area depends on the value of bitfield HDPEXT_ACCDIS[7:0]:

- If it equals 0xA3 any value can be written in HDP_EXT_END[11:0].
- If it equals 0x5C the value stored in HDP_EXT_END[11:0] can only be increased (writing a lower value is ignored).
- For any other value of HDPEXT_ACCDIS any writes to FLASH_xHDPEXTR is ignored.

Note: Out of reset HDP extension is not active.

5.6 FLASH low-power modes

5.6.1 Introduction

Table 35 summarizes the behavior of the embedded flash memory in microcontroller low-power modes. The embedded flash memory belongs to the Core domain.

Table 35. Effect of low-power modes on the embedded flash memory

Power mode System state	Core domain voltage range	Allowed if FLASH busy	FLASH power mode (in Core domain)
Run	VOS0,1,2	Yes	Run
Stop	SVOS low / high	No	Clock gated or Stopped
Standby	Off	No	Off

When the system state changes, or within a given system state, the embedded flash memory might be subjected to a different voltage supply range (VOS) depending on the application. The procedure to switch the embedded flash memory into various power modes (run, clock gated, stopped, off) is described below.

Note: For more information on microcontroller power states, refer to the Power control section (PWR).

5.6.2 Managing the FLASH domain switching to Stop or Standby

As explained in *Table 35*, if the embedded flash memory informs the reset and clock controller (RCC) that it is busy (that is, BUSY, QW, WBNE is set), the microcontroller cannot switch the Core domain to Stop or Standby mode.

Note: CRC_BUSY is not taken into account.

There are two ways to release the embedded flash memory:

- For each bank x, reset the WBNE busy flag in the FLASH_xSR register by any of the following actions:
  - Complete the write buffer with missing data.
  - Force the write operation without filling-in the missing data, by activating the FW bit in the FLASH_CR register. This forces all missing data “high”.
  - Reset the PG bit in the FLASH_CR register. This disables the write buffer and consequently leads to the loss of its content.
- Poll the QW busy bits in the FLASH_xSR register until they are cleared. This indicates that all recorded write, erase and option change operations are complete in both banks.

The microcontroller can then switch the domain to Stop or Standby mode.

5.7 FLASH error management

The embedded flash memory automatically reports when an error occurs during a read, program or erase operation. A wide range of errors are reported:

- *Write protection error (WRPERRF)*

- *Programming sequence error (PGSERRF)*
- *Strobe error (STRBERRF)*
- *Inconsistency error (INCERRF)*
- *Error correction code error (SNECCERRF/DBECCERRF)*
- *Read secure error (RDSERRF)*
- *CRC read error (CRCRDERRF)*
- *Option byte change error (OPTERRF)*
- *Option byte loading error (OBLERRF)*
- *Option byte key error flag (OBKERRF)*

The application software can individually enable the interrupt for each error, as detailed in *Section 5.8: FLASH interrupts*.

Note: For some errors, the application software must clear the error flag before attempting a new operation.

Note: General-purpose errors (for example INCERRF) are taken into account when queues are loaded.
Temporal isolation and other protection errors (for example WRPERRF, RDSERRF) are taken into account at FSM level, in front of the flash bank interface controller.

5.7.1 Write protection error (WRPERRF)

When an illegal erase/program operation is attempted to the nonvolatile memory of bank x, the embedded flash memory sets the write protection error flag WRPERRF in FLASH_xISR register.

An erase operation is rejected and flagged as illegal if it targets one of the following memory areas:

- A page belonging to a valid HDP user flash area (even partially).
- A page write-locked with WRPG
- The read-only/OTP area

A program operation is ignored and flagged as illegal if it targets one of the following memory areas:

- A user page currently protected by HDP (see *Section 5.5.4*)
- A user page write-locked with WRPG
- An OTP block, locked with OTPL
- A read-only section
- The system flash memory (protected or not)
- A reserved area

When the WRPERRF flag is raised, the operation is rejected and nothing is changed in the corresponding bank. If a write burst operation was ongoing, WRPERRF is raised each time a flash word write operation is processed by the embedded flash memory.

Note: The WRPERRF flag does not block any new erase/program operation.

Not clearing the WRPERRF flag does not generate a programming sequence error (see below).

The WRPERRF flag is cleared by setting the corresponding bit in the FLASH_xICR register.

If the WRPERRIE bit in the FLASH_xIER register is set, an interrupt is generated when the WRPERRF flag is raised (see *Section 5.8: FLASH interrupts* for details).

5.7.2 Programming sequence error (PGSERRF)

When the programming sequence is incorrect in bank x, the embedded flash memory sets the programming sequence error flag PGSERRF in FLASH_xISR register.

More specifically, the PGSERRF flag is set if one of the conditions below is met:

- A write operation is requested but the program enable bit (PG or PG_OTP) has not been set in the FLASH_xCR register prior to the request. This program enable bit must stay at 1 until write is added to the operation queue (QW rises).
- The inconsistency error (INCERRF) has not been cleared before requesting a new write operation.

When the PGSERRF flag is raised, the current program operation is aborted and nothing is changed in the corresponding bank. The corresponding write data buffer is also flushed. If a write burst operation was ongoing, PGSERRF is raised at the end of the burst.

Note: When the PGSERRF flag is raised, there is a risk that the last write operation performed by the application has been lost because of the above protection mechanism. Hence it is recommended to generate interrupts on PGSERRF and verify in the interrupt handler if the last write operation has been successful by reading back the value in the flash memory.

The PGSERRF flag also blocks any new program operation. This means that PGSERRF must be cleared before starting a new program operation on the corresponding bank.

The PGSERRF flag is cleared by setting the corresponding bit in the FLASH_xICR register.

If the PGSERRIE bit in FLASH_xIER register is set, an interrupt is generated when the PGSERRF flag is raised. See *Section 5.8: FLASH interrupts* for details.

5.7.3 Strobe error (STRBERRF)

When the application software writes several times to the same byte write buffer toward the bank x, the embedded flash memory sets the strobe error flag STRBERRF in the FLASH_xISR register.

When the STRBERRF flag is raised, the current program operation is not aborted and new byte data replaces the old data. The application can ignore the error, proceed with the current write operation and request new write operations. If a write burst was ongoing, STRBERRF is raised at the end of the burst.

The STRBERRF flag is cleared by setting the corresponding bit in the FLASH_xICR register.

If the STRBERRIE bit in the FLASH_xIER register is set, an interrupt is generated when the STRBERRF flag is raised. See *Section 5.8: FLASH interrupts* for details.

5.7.4 Inconsistency error (INCERRF)

When a programming inconsistency is detected to bank x, the embedded flash memory sets the inconsistency error flag INCERRF in the FLASH_xISR register.

More specifically, the INCERRF flag is set when one of the following conditions is met:

- A write operation is attempted before completion of the previous write operation, for example:
  - The application software starts a write operation to fill the 128-bit write buffer, but sends a new write burst request to a different flash memory address before the buffer is full.
  - One master starts a write operation, but before the buffer is full, another master starts a new write operation to the same address or to a different address.
- A wrap burst request issued by a master overlaps two or more 128-bit flash-word addresses, that is, wrap bursts must be done within 128-bit flash-word address boundaries.

Following the sequence below is recommended to avoid losing data when an inconsistency error occurs:

1. Execute a handler routine when the INCERRF flag is raised.
2. Stop all write requests to embedded flash memory.
3. Verify that the write operations that have been requested just before the INCERRF event have been successful by reading back the programmed values from the memory.
4. Clear the INCERRF bit in the FLASH_xICR register.
5. Restart the write operations where they have been interrupted.

Note: The INCERRF flag must be cleared before starting a new write operation, otherwise a sequence error (PGSERRF) is raised.

Any write triggering inconsistency error is discarded by embedded flash memory.

5.7.5 Error correction code error (SNECCERRF/DBECCERRF)

When a single error correction is detected during a read to the bank x the embedded flash memory sets the single error correction flag SNECCERRF in the FLASH_xISR register.

When two ECC errors are detected during a read to the bank x, the embedded flash memory sets the double error detection flag DBECCERRF in the FLASH_xISR register.

When the SNECCERRF flag is raised, the corrected read data are returned. Hence the application can ignore the error and request new read operations.

If a read burst operation was ongoing, the SNECCERRF or DBECCERRF flag is raised each time a new data is sent back to the requester through the corresponding system bus.

When the SNECCERRF (or DBECCERRF respectively) flag is raised, the address of the flash word that generated the error is saved in the FLASH_xECCSFAR (or FLASH_xECCDFAR respectively) register. This register is automatically cleared when the associated flag that generated the error is cleared.

Note: In the case of successive single correction (or double detection respectively) errors, only the address corresponding to the first error is stored in the *FLASH_xECCSFAR (or FLASH_xECCDFAR respectively) register.*

When the DBECCERRF flag is raised reading the user flash memory, a bus error is generated. In the case of successive double error detections, a bus error is generated each time a new data is sent back to the requester through the AXI interface.

Note: It is not mandatory to clear the *SNECCERRF or DBECCERRF flags before starting a new read operation.*

The SNECCERRF and DBECCERRF flags are cleared by setting the corresponding bit in the FLASH_xICR register.

If the SNECCERRIE (or respectively DBECCERRIE) bit in the FLASH_xISR register is set, an interrupt is generated when the SNECCERRF (or respectively DBECCERRF) flag is raised. See *Section 5.8: FLASH interrupts* for details.

5.7.6 Read secure error (RDSERRF)

When an illegal read or execute operation is attempted to a hide protected area of bank x, the embedded flash memory sets the read secure the error flag RDSERRF in the FLASH_xISR register. For more information on illegal access definition, refer to *Section 5.5.4: Hide protection (HDP)*.

When the RDSERRF flag is raised, the current read operation is aborted and the application can request new read operations. If a read burst was ongoing, RDSERRF is raised each time a data is sent back to the requester through the AXI interface.

Note: The bus error is raised only if the illegal access is due to an instruction fetch.

The RDSERRF flag is cleared by setting the corresponding bit in the FLASH_xICR register.

If the RDSERRIE bit in the FLASH_xIER register is set, an interrupt is generated when the RDSERRF flag is raised (see *Section 5.8: FLASH interrupts* for details).

5.7.7 CRC read error (CRCRDERRF)

After a CRC computation in bank x, the embedded flash memory sets the CRC read error flag CRCERR in the FLASH_xISR register when one or more address belonging to a protected area was read by the CRC module. A protected area corresponds to a valid HDP area in user flash, for which any read is currently illegal (see *Section 5.5.4* for details).

The CRCRDERRF flag is raised when the CRCEND bit is set (end of CRC calculation). In this case, it is likely that the CRC result is wrong since illegal read operations to protected areas return null values.

The CRCRDERRF flag is cleared by setting the corresponding bit in the FLASH_xICR register.

If the CRCRDERRIE bit in the FLASH_xIER register is set, an interrupt is generated when the CRCRDERRF flag is raised together with the CRCEND bit (see *Section 5.8: FLASH interrupts* for details).

5.7.8 Option byte change error (OPTERRF)

When the embedded flash memory finds an error during an option change operation, it aborts the operation and sets the option byte change the error flag OPTERRF in the FLASH_OPTISR register. Refer to *Section 5.4.3: Option byte words modification* for details.

The OPTERRF flag is cleared by setting the corresponding bit in the FLASH_OPTICR register.

If the OPTERRIE bit in the FLASH_OPTCR register is set, an interrupt is generated when the OPTERRF flag is raised (see *Section 5.8: FLASH interrupts* for details).

It is mandatory to clear the OPTERRF flag before starting a new option byte change, or an option byte key programming.

5.7.9 Option byte loading error (OBLERRF)

OBLERRF error flag is set in the FLASH_xISR register when a double error detection was found when loading one or more option bytes during an option byte loading sequence (see *Section 5.4.2*). In this case, the application should verify the correctness of the option byte information comparing FLASH_xxSR and FLASH_xxSRP registers.

The OBLERRF flag is cleared by setting the corresponding bit in the FLASH_xICR register. If the OBLERRIE bit in the FLASH_xIER register is set, an interrupt is generated when the OBLERRF flag is raised (see *Section 5.8: FLASH interrupts* for details).

5.7.10 Option byte key error flag (OBKERRF)

The OBKERRF error flag is set in the FLASH_OPTISR register in two cases:

- A double error detection was found when loading the requested option byte key. In this case, if this key is provisioned again the error should disappear.
- The application tried to program a nonvirgin option byte key, or the key corresponding to OBKINDEX[4:0] was not requested to be erased using FLASH_OBKCLR register.
- The application tried to program and erase keys at the same time (KEYPROG and KEYERASE set when KEYSTART is set).

It is mandatory to clear the OBKERRF flag before doing any operation on option byte keys because when OBKERRF is set, writes to the KEYSTART bit in the FLASH_OBKCR are ignored.

The OBKERRF flag is cleared by setting the corresponding bit in the FLASH_OPTICR register.

If the OBKERRIE bit in the FLASH_OPTCR register is set, an interrupt is generated when the OBKERRF flag is raised (see *Section 5.8: FLASH interrupts* for details).

5.7.11 Miscellaneous HardFault errors

The following events generate a bus error on the corresponding bus interface:

- On the AXI system bus:
  - Illegal instruction fetch to HDP protected area because of the life cycle (e.g. protected system flash when NVSTATE = NVOPEN).
- On AHB configuration or system bus:
  - Wrong key input to FLASH_KEYR or FLASH_OPTKEYR
  - 8-bit accesses to system AHB interface

5.8 FLASH interrupts

The embedded flash memory can generate a maskable interrupt to signal the following events on a given bank:

- Read and write errors (see *Section 5.7: FLASH error management*)
  - Single ECC error correction during read operation
  - Double ECC error detection during read operation
  - Write inconsistency error
  - Bad programming sequence
  - Strobe error during write operations
  - Option change error
  - Option byte keys error
- Security errors (see *Section 5.7: FLASH error management*)
  - Write protection error
  - Read security error
  - CRC computation on HDP area error

Option byte loading error

- Miscellaneous events (described below)
  - End of programming
  - CRC computation complete

These multiple sources are combined into a single interrupt signal, flash_it, which is the only interrupt signal from the embedded flash memory that drives the NVIC (nested vectored interrupt controller).

The user can individually enable or disable embedded flash memory interrupt sources by changing the mask bits in the FLASH_xIER and OPTCR registers. Setting the appropriate mask bit enables the interrupt.

Note: Prior to writing, the FLASH_xIER register must be unlocked as explained in Section 5.5.1: FLASH configuration protection

Table 36 gives a summary of the available embedded flash memory interrupt features.

Note: Some flags need to be cleared before a new operation is triggered.

Table 36. Flash interrupt request

Interrupt event	Event flag	Enable control bit	Clear flag to resume operation	Bus error
End-of-program event	EOPF	EOPIE	N/A	N/A
CRC complete event	CRCENDF	CRCENDIE	N/A	N/A
Write protection error	WRPERRF	WRPERRIE	No	No
Programming sequence error	PGSERRF	PGSERRIE	**Yes**(1)	No
Strobe error	STRBERRF	STRBERRIE	No	No
Inconsistency error	INCERRF	INCERRIE	**Yes**(1)	No
ECC single error correction event	SNECCERRF	SNECCERRIE	No	No

Table 36. Flash interrupt request

Interrupt event	Event flag	Enable control bit	Clear flag to resume operation	Bus error
ECC double error detection event	DBECCERRF	DBECCERRIE	No	**Yes**(2)
Read secure error	RDSERRF	RDSERRIE	No	No (data) **Yes** (fetch)
CRC error	CRCRDERRF	CRCRDERRIE	No	No

1. Programming still possible on the AXI interface that did not trigger the inconsistency error, on the flash bank that does not have the flag bit raised. See *Section 5.7.4: Inconsistency error (INCERRF)* for details.
2. AXI bus only

The status of the individual maskable interrupt sources described in *Table 36* (except for option byte error) can be read from the FLASH_xISR registers. They can be cleared by setting the corresponding bit in the FLASH_xICR register.

No unlocking mechanism is required to clear an interrupt.

End-of-program event

Setting the end-of-operation interrupt enable bit (EOPIE) in the FLASH_xIER register enables the generation of an interrupt at the end of an erase operation, a program operation or an option byte change. The EOPF bit in the FLASH_xISR register is also set when one of these events occurs.

Setting the EOPF bit in the FLASH_xICR register clears the EOPF flag in the FLASH_xISR register.

CRC end of calculation event

Setting the CRC end-of-calculation interrupt enable bit (CRCENDIE) in the FLASH_xIER register enables the generation of an interrupt at the end of a CRC operation. The CRCENDF bit in the FLASH_xISR register is also set when this event occurs.

Setting the CRCENDF bit in the FLASH_xICR register clears the same flag in the FLASH_xISR register.

5.9 FLASH registers

5.9.1 FLASH access control register (FLASH_ACCR)

Address offset: 0x000

Reset value: 0x0000 0013

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

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

Bits 5:4 **WRHIGHFREQ[1:0]**: Flash signal delay

These bits are used to control the delay between nonvolatile memory signals during programming operations. Application software has to program them to the correct value depending on the embedded flash memory interface frequency. Refer to *Table 25* for details.

Note: Embedded flash does not verify that the configuration is correct.

Bits 3:0 **LATENCY[3:0]**: Read latency

These bits are used to control the number of wait states used during read operations on both nonvolatile memory banks. The application software has to program them to the correct value depending on the embedded flash memory interface frequency and voltage conditions. Refer to *Table 25* for details.

0x0: Zero wait state used to read a word from nonvolatile memory

0x1: One wait state used to read a word from nonvolatile memory

0x2: Two wait states used to read a word from nonvolatile memory

...

0xF: Fifteen wait states used to read from nonvolatile memory

Note: Embedded flash does not verify that the configuration is correct.

5.9.2 FLASH access control register (FLASH_ACCR_ALIAS)

Address offset: 0x100

Reset value: 0x0000 0013

This register is an alias of FLASH_ACCR.

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

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

Bits 5:4 **WRHIGHFREQ[1:0]**: Flash signal delay

These bits are used to control the delay between nonvolatile memory signals during programming operations. Application software has to program them to the correct value depending on the embedded flash memory interface frequency. Refer to *Table 25* for details.

Note: Embedded flash does not verify that the configuration is correct.

Bits 3:0 **LATENCY[3:0]**: Read latency

These bits are used to control the number of wait states used during read operations on both nonvolatile memory banks. The application software has to program them to the correct value depending on the embedded flash memory interface frequency and voltage conditions. Refer to *Table 25* for details.

0x0: Zero wait state used to read a word from nonvolatile memory

0x1: One wait state used to read a word from nonvolatile memory

0x2: Two wait states used to read a word from nonvolatile memory

...

0xF: Fifteen wait states used to read from nonvolatile memory

Note: Embedded flash does not verify that the configuration is correct.

5.9.3 FLASH bank x control key register (FLASH_xKEYR)

Address offset: 0x004 + 0x100 * (x-1), (x = A to B)

Reset value: 0x0000 0000

This register is impacted by the swapping function (see *Section 5.3.14*).

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

Bits 31:0 **CUKEY[31:0]**: Control unlock key

Following values must be written to FLASH_xKEYR consecutively to unlock FLASH_xCR register:

1st key = 0x4567 0123

2nd key = 0xCDEF 89AB

Reads to this register returns zero. If above sequence is wrong or performed twice, the FLASH_xCR register is locked until the next system reset, and access to it generates a bus error.

5.9.4 FLASH bank x control register (FLASH_xCR)

Address offset: 0x010 + 0x100 * (x-1), (x = A to B)

Reset value: 0x0000 0001

When the LOCK bit in this register is set writes to all other bits are ignored.

Write access to this register is controlled by FLASH_KEYR. This register is impacted by the swapping function (see *Section 5.3.14*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_EN	PG_OTP
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PSN[6:0]							START	FW	BER	PER	PG	LOCK
			rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw	rs

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

Bit 17 **CRC_EN**: CRC enable

Setting this bit enables the CRC calculation. CRC_EN does not start CRC calculation but enables CRC configuration through FLASH_xCRCCR register.

When CRC_BUSY flag is cleared in FLASH_xSR, CRC calculation is disabled by clearing CRC_EN bit. Doing so sets CRCDATA to 0x0, clears CRC configuration and resets the content of FLASH_xCRCDATAR register.

Bit 16 **PG_OTP**: Program enable for OTP area

Set this bit to enable write operations to OTP area.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:6 **PSN[6:0]**: Page erase selection number

These bits are used to select the target page for an erase operation in the bank x (they are unused otherwise).

0x0: Page 0 selected

0x1: Page 1 selected

...

0x3F: Page 63 selected

0x40: Page 64 selected *(STM32H7PxxI devices only)*

...

0x7F: Page 127 selected *(STM32H7PxxI devices only)*

Bit 5 **START**: Erase start control bit

This bit is used to start a page erase or a bank erase operation in the bank x. The embedded flash memory resets START when the corresponding operation has been acknowledged. The user application cannot access any embedded flash memory register until the operation is acknowledged.

Bit 4 **FW**: Force write

This bit forces a write operation in the bank x even if the write buffer is not full. In this case all bits not written are set by hardware. FW reads as zero.

Note: Using a force-write operation prevents the application from updating later the missing bits with something other than 1, because this would likely to lead to a permanent ECC error.

When FW is set when write buffer is empty no programming is done (BUSY and EOP stays clear).

Bit 3 **BER**: Bank erase request

Setting this bit requests a bank erase operation in the bank x (user flash memory only).
Write protection error is triggered when a bank erase is required and some pages are protected.
BER has a higher priority than PER: if both are set, the embedded flash memory executes a bank erase.
0: Bank erase is not requested
1: Bank erase is requested. Actual erase is started setting START bit in this register.

Bit 2 **PER**: Page erase request

Setting this bit requests a page erase in the bank x.
Write protection error is triggered when a page erase is required on at least one protected page.
BER has a higher priority than PER: if both bits are set, the embedded flash memory executes a bank erase.
0: Page erase not requested
1: Page erase requested

Bit 1 **PG**: Programming control bit

Setting this bit allows write operations in bank x. In the bank x, when PG is cleared, the internal buffer is cleared.
0: Programming disabled
1: Programming enabled

Bit 0 **LOCK**: Configuration lock bit

When this bit is set write to all other bits in this register, and to FLASH_xIER register, are ignored.
Clearing this bit requires the correct write sequence to FLASH_xKEYR register (see this register for details). If a wrong sequence is executed, or if the unlock sequence is performed twice, this bit remains locked until the next system reset.
During the write access to set LOCK bit from 0 to 1, it is possible to change the other bits of this register.
0: FLASH_xCR and FLASH_xIER registers are unlocked
1: Writes to FLASH_xIER, and to other bits than LOCK in FLASH_xCR, are ignored

5.9.5 FLASH bank x status register (FLASH_xSR)

Address offset: 0x014 + 0x100 * (x-1), (x = A to B)

Reset value: 0x0000 0000

Some bits in this register are impacted by the swapping function (see *Table 29* in *Section 5.3.14*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWAP_BANK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IS_OPTCHANGE	IS_ERASE	IS_PROGRAM	CRC_BUSY	QW	WBNE	BUSY
									r	r	r	r	r	r	r

Bit 31 **SWAP_BANK**: Swapped bank

This bit is set following a power-on-reset if SWAP_BANK is set in FLASH_OBW1SR. It controls the physical bank allocation to the bank x.

0: Physical bank 1 is allocated to the bank if x = A (bank 2 if x = B)

1: Physical bank 2 is allocated to the bank if x = A (bank 1 if x = B)

Bits 30:7 Reserved, must be kept at reset value.

Bit 6 **IS_OPTCHANGE**: Is an option change

This bit is set together with BUSY when an option change operation is ongoing in the bank x. It is cleared when BUSY is cleared.

This flag can also raise with IS_PROGRAM or IS_ERASE, because a program or erase step is ongoing during option change.

Bit 5 **IS_ERASE**: Is an erase

This bit is set together with BUSY when an erase operation is ongoing in the bank x. It is cleared when BUSY is cleared.

This flag can also raise with IS_PROGRAM or IS_OPTCHANGE, because different operations can run in parallel on different banks or because an erase operation can happen during an option change.

Bit 4 **IS_PROGRAM**: Is a program

This bit is set together with BUSY when a program operation is ongoing in the bank x. It is cleared when BUSY is cleared.

This flag can also raise with IS_ERASE or IS_OPTCHANGE, because different operations can run in parallel on different banks or because an program operation can happen during an option change.

Bit 3 **CRC_BUSY**: CRC busy flag

This bit is set when a CRC calculation is ongoing in the bank x. This bit cannot be forced to 0. The user must wait until the CRC calculation has completed or disable CRC computation using CRC_EN bit in FLASH_xCR register.

0: No CRC calculation ongoing

1: CRC calculation ongoing

Bit 2 **QW**: Wait queue flag

This bit is set when a write, erase or option byte change operation (including option byte key change) is pending in the command queue buffer of this bank x. It is not possible to know what type of programming operation is present in the queue.

This flag is reset by hardware when all write, erase or option byte change operations have been executed and thus removed from the waiting queue(s). This bit cannot be forced to 0. It is reset after a deterministic time if no other operations are requested.

0: No write, erase or option byte change operations waiting in the operation queues

1: At least one write, erase or option byte change operation is waiting in the operation queue

Bit 1 **WBNE**: Write buffer not empty flag

This bit is set when the embedded flash memory is waiting for new data to complete the write buffer. In this state, the write buffer is not empty. WBNE is reset by hardware each time the write buffer is complete or the write buffer is emptied following one of the event below:

- the application software forces the write operation using FW bit in FLASH_xCR
- the embedded flash memory detects an error that involves data loss
- the application software has disabled write operations in this bank

This bit cannot be forced to 0. To reset it, clear the write buffer by performing any of the above listed actions, or send the missing data.

0: Write buffer empty or full

1: Write buffer waiting data to complete

Bit 0 **BUSY**: Busy flag

This bit is set when an effective write, erase or option byte change operation is ongoing on this bank x. It is possible to know what type of operation is being executed reading the flags IS_PROGRAM, IS_ERASE and IS_OPTCHANGE.

BUSY cannot be cleared by application. It is automatically reset by hardware every time a step in a write, erase or option byte change operation completes. It is not recommended to do software polling on BUSY to know when one operation completed because, depending of operation, more pulses are possible for one only operation. For software polling it is therefore better to use QW flag or to check the EOPF flag.

0: No programming, erase or option byte change operation are being executed

1: Programming, erase or option byte change operation are being executed. See flags IS_PROGRAM, IS_ERASE and IS_OPTCHANGE for details.

5.9.6 FLASH bank x interrupt enable register (FLASH_xIER)

Address offset: 0x020 + 0x100 * (x-1), (x = A to B)

Reset value: 0x0000 0000

Writes to this register are ignored if the LOCK bit is set in the FLASH_CR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	CRCRDERRIE	CRCENDIE	DBECCERRIE	SNECCERRIE	RDSERRIE	Res.	Res.	INCERRIE	OBLERRIE	STRBERRIE	PGSERRIE	WRPERRIE	EOPIE
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **CRCRDERRIE**: CRC read error interrupt enable bit
0: No interrupt is generated when CRCRDERRF bit is set in FLASH_xISR register
1: An interrupt is generated when CRCRDERRF bit is set in FLASH_xISR register

Bit 27 **CRCENDIE**: CRC end of calculation interrupt enable bit
0: No interrupt is generated when CRCEN bit is set in FLASH_xISR register
1: An interrupt is generated when CRCEN bit is set in FLASH_xISR register

Bit 26 **DBECCERRIE**: ECC double detection error interrupt enable bit
0: No interrupt is generated when DBECCERRF bit is set in FLASH_xISR register
1: An interrupt is generated when DBECCERRF bit is set in FLASH_xISR register

Bit 25 **SNECCERRIE**: ECC single correction error interrupt enable bit
0: No interrupt is generated when SNECCERRF bit is set in FLASH_xISR register
1: An interrupt is generated when SNECCERRF bit is set in FLASH_xISR register

Bit 24 **RDSERRIE**: Read security error interrupt enable bit
0: No interrupt is generated when RDSERRF bit is set in FLASH_xISR register
1: An interrupt is generated when RDSERRF bit is set in FLASH_xISR register

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **INCERRIE**: Inconsistency error interrupt enable bit
0: No interrupt is generated when INCERRF bit is set in FLASH_xISR register
1: An interrupt is generated when INCERRF bit is set in FLASH_xISR register

Bit 20 **OBLERRIE**: Option byte loading error interrupt enable bit
0: No interrupt is generated when OBLERRF bit is set in FLASH_xISR register
1: An interrupt is generated when OBLERRF bit is set in FLASH_xISR register

Bit 19 **STRBERRIE**: Strobe error interrupt enable bit
0: No interrupt is generated when STRBERRF bit is set in FLASH_xISR register
1: An interrupt is generated when STRBERRF bit is set in FLASH_xISR register

Bit 18 **PGSERRIE**: Programming sequence error interrupt enable bit
0: No interrupt is generated when PGSERRF bit is set in FLASH_xISR register
1: An interrupt is generated when PGSERRF bit is set in FLASH_xISR register

Bit 17 **WRPERRIE**: Write protection error interrupt enable bit
0: No interrupt is generated when WRPERRF bit is set in FLASH_xISR register
1: An interrupt is generated when WRPERRF bit is set in FLASH_xISR register

Bit 16 **EOPIE**: End-of-program interrupt control bit
0: No interrupt is generated when OEPF bit is set in FLASH_xISR register
1: An interrupt is generated when OEPF bit is set in FLASH_xISR register

Bits 15:0 Reserved, must be kept at reset value.

5.9.7 FLASH bank x interrupt status register (FLASH_xISR)

Address offset: 0x024 + 0x100 * (x-1), (x = A to B)

Reset value: 0x0000 0000

Some bits in this register are impacted by the swapping function (see *Table 29* in *Section 5.3.14*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	CRCRDERRF	CRCENDF	DBECCERRF	SNECCERRF	RDSERRF	Res.	Res.	INCERRF	OBLERRF	STRBERRF	PGSERRF	WRPERRF	EOPF
			r	r	r	r	r			r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **CRCRDERRF**: CRC read error flag

This bit is set when a word is found read protected during a CRC operation. An interrupt is generated if CRCRDIE is set. Setting CRCRDERRF bit in FLASH_ICR register clears this bit.

0: No protected area detected inside addresses read by CRC

1: At least one protected area has been detected inside address read by CRC. As a result CRC result is very likely incorrect.

This flag is valid only when CRCEND bit is set.

Bit 27 **CRCENDF**: CRC end flag

This bit is set when the CRC computation has completed. An interrupt is generated if CRCENDIE is set. It is not necessary to reset CRCEND before restarting CRC computation. Setting CRCENDF bit in FLASH_ICR register clears this bit.

0: CRC computation not complete

1: CRC computation complete

Bit 26 **DBECCERRF**: ECC double error flag

This bit is set when an ECC double detection error occurs during a read operation. An interrupt is generated if DBECCERRIE is set. Setting DBECCERRF bit in FLASH_ICR register clears this bit.

0: No ECC double detection error occurred

1: ECC double detection error occurred

Bit 25 **SNECCERRF**: ECC single error flag

This bit is set when an ECC single correction error occurs during a read operation. An interrupt is generated if SNECCERRIE is set. Setting SNECCERRF bit in FLASH_ICR register clears this bit.

0: No ECC single correction error occurred

1: ECC single correction error occurred

Bit 24 **RDSERRF**: Read security error flag

This bit is set when a read security error occurs (read access to hide protected area with incorrect hide protection level). An interrupt is generated if RDSERRIE is set. Setting RDSERRF bit in FLASH_ICR register clears this bit.

0: No security error occurred

1: A security error occurred

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **INCERRF:** Inconsistency error flag

This bit is set when a inconsistency error occurs. An interrupt is generated if INCERRIE is set. Setting INCERRF bit in the FLASH_ICR register clears this bit.

0: No inconsistency error occurred

1: A inconsistency error occurred

Bit 20 **OBLERRF:** Option byte loading error flag

This bit is set when an error is found during the option byte loading sequence. An interrupt is generated if OBLERRIE is set.

Setting OBLERRF bit in the FLASH_ICR register clears this bit.

0: No error found during option byte loading sequence

1: Some errors found during option byte loading sequence

Bit 19 **STRBERRF:** Strobe error flag

This bit is set when a strobe error occurs (when the master attempts to write several times the same byte in the write buffer). An interrupt is generated if the STRBERRIE bit is set. Setting STRBERRF bit in FLASH_ICR register clears this bit.

0: No strobe error occurred

1: Astrobe error occurred

Bit 18 **PGSERRF:** Programming sequence error flag

This bit is set when a sequence error occurs. An interrupt is generated if the PGSERRIE bit is set. Setting PGSERRF bit in FLASH_ICR register clears this bit.

0: No sequence error occurred

1: Asequence error occurred

Bit 17 **WRPERRF**: Write protection error flag

This bit is set when a protection error occurs during a program operation. An interrupt is also generated if the WRPERRIE is set. Setting WRPERRF bit in FLASH_ICR register clears this bit.

0: No write protection error occurred

1: A write protection error occurred

Bit 16 **EOPF**: End-of-program flag

This bit is set at the end of an erase operation, a program operation or an option byte change. An interrupt is generated if the EOPIE is set. It is not necessary to reset EOPF before starting a new operation. Setting EOPF bit in FLASH_ICR register clears this bit.

0: No programming or erase operation completed

1: Programming, or erasing operation completed. Possible option byte change completion if x = A (SWAP_BANK cleared in FLASH_xSR) or x = B (SWAP_BANK set).

Bits 15:0 Reserved, must be kept at reset value.

5.9.8 FLASH bank x interrupt clear register (FLASH_xICR)

Address offset: 0x028 + 0x100 * (x-1), (x = A to B)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	CRCRDERRF	CRCENDF	DBECCERRF	SNECCERRF	RDSERRF	Res.	Res.	INCERRF	OBLERRF	STRBERRF	PGSERRF	WRPERRF	EOPF
			w	w	w	w	w			w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **CRCRDERRF**: CRC error flag clear
Setting this bit clears CRCRDERRF flag in FLASH_ISR register.

Bit 27 **CRCENDF**: CRC end flag clear
Setting this bit clears CRCENDF flag in FLASH_ISR register.

Bit 26 **DBECCERRF**: ECC double error flag clear
Setting this bit clears DBECCERRF flag in FLASH_ISR register. If the SNECCERRF flag of FLASH_ISR register is also cleared, FLASH_ECCDFADDR register is reset to zero as well.

Bit 25 **SNECCERRF**: ECC single error flag clear
Setting this bit clears SNECCERRF flag in FLASH_ISR register. If the DBECCERRF flag of FLASH_ISR register is also cleared, FLASH_ECCSFADDR register is reset to zero as well.

Bit 24 **RDSERRF**: Read security error flag clear
Setting this bit clears RDSERRF flag in FLASH_ISR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **INCERRF**: Inconsistency error flag clear
Setting this bit clears INCERRF flag in FLASH_ISR register.

Bit 20 **OBLERRF**: Option byte loading error flag clear
Setting this bit clears OBLERRF flag in FLASH_ISR register.

Bit 19 **STRBERRF**: Strobe error flag clear
Setting this bit clears STRBERRF flag in FLASH_ISR register.

Bit 18 **PGSERRF**: Programming sequence error flag clear
Setting this bit clears PGSERRF flag in FLASH_ISR register.

Bit 17 **WRPERRF**: Write protection error flag clear
Setting this bit clears WRPERRF flag in FLASH_ISR register.

Bit 16 **EOPF**: End-of-program flag clear
Setting this bit clears EOPF flag in FLASH_ISR register.

Bits 15:0 Reserved, must be kept at reset value.

5.9.9 FLASH bank x CRC control register (FLASH_xCRCCR)

Address offset: 0x030 + 0x100 * (x-1), (x = A to B)

Reset value: 0x0018 0000

Writes to this register are ignored if the CRC_EN bit is cleared in FLASH_xCR or if CRC_BUSY is set in FLASH_xSR. Reads to this registers return default values when the

CRC_EN bit is cleared. This register is impacted by the swapping function (see *Section 5.3.14*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALL_BANK	CRC_BURST[1:0]		Res.	Res.	CLEAN_CRC	START_CRC
									w	rw	rw			w	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CLEAN_PAGE	ADD_PAGE	CRC_BY_PAGE	Res.	Res.	CRC_PAGE[6:0]						
				w	w	rw			rw	rw	rw	rw	rw	rw	rw

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

Bit 22 **ALL_BANK**: All pages in bank selection

Set this write-only bit to add all pages of bank x to the list of pages on which the CRC is calculated. Reading this bit returns zero.

Bits 21:20 **CRC_BURST[1:0]**: CRC burst size

CRC_BURST bits set the size of the bursts that are generated by the CRC calculation unit. A flash word is 128-bit.

00: every burst has a size of 4 flash words (64 Bytes)

01: every burst has a size of 16 flash words (256 Bytes)

10: every burst has a size of 64 flash words (1 Kbytes)

11: every burst has a size of 256 flash words (4 Kbytes)

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CLEAN_CRC**: CRC clear bit

Setting CLEAN_CRC to 1 clears the current CRC result stored in the FLASH_CRCDATAR register.

Bit 16 **START_CRC**: CRC start bit

START_CRC bit triggers a CRC calculation in the bank x using the current configuration. No CRC calculation can launched when an option byte change operation is ongoing because all read accesses to embedded flash memory registers are put on hold until the option byte change operation has completed.

This bit is cleared when CRC computation starts.

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

Bit 11 **CLEAN_PAGE**: CRC page list clear bit

When this bit is set the list of pages on which the CRC is calculated is cleared.

Bit 10 **ADD_PAGE**: CRC page select bit

When this bit is set the page in this bank x whose number is written in CRC_PAGE is added to the list of pages on which the CRC is calculated.

Bit 9 **CRC_BY_PAGE**: CRC page mode select bit

When this bit is set the CRC calculation is performed in the bank x at page level, on the pages present in the list of pages. To add a page to this list, use ADD_PAGE and CRC_PAGE bits. To clean the list, use CLEAN_PAGE bit.

When CRC_BY_PAGE is cleared the CRC calculation is performed on all addresses defined between start and end addresses defined in FLASH_CRCSADDR and FLASH_CRCEADDR registers.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **CRC_PAGE[6:0]**: CRC page number

CRC_PAGE is used to select one user flash page of bank x to be added to the list of pages on which the CRC is calculated. The CRC can be computed either between two addresses (using registers FLASH_xCRCSADDR and FLASH_xCRCEADDR) or on a list of pages using this register. If this latter option is selected, it is possible to add a page to the list of pages by programming the page number in CRC_PAGE and then setting ADD_PAGE bit.

The list of pages can be erased either by setting CLEAN_PAGE bit or by disabling the CRC computation.

0x0: Page 0 for CRC

0x1: Page 1 for CRC

...

0x3F: Page 63 for CRC

0x40: Page 64 for CRC *(STM32H7PxxI devices only)*

...

0x7F: Page 127 for CRC *(STM32H7PxxI devices only)*

5.9.10 FLASH bank x CRC start address register (FLASH_xCRCSADDR)

Address offset: 0x034 + 0x100 * (x-1), (x = A to B)

Reset value: 0x0000 0000

Writes to this register are ignored if the CRC_BUSY bit is set in the FLASH_xSR register, or CRC_EN bit is set in FLASH_xCR register.

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:2 **CRC_START_ADDR[19:2]**: CRC start address

This register is used when CRC_BY_PAGE is cleared. It must be programmed to the address of the start address on which the CRC calculation is done on bank x.

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

5.9.11 FLASH bank x CRC end address register (FLASH_xCRCEADDR)

Address offset: 0x038 + 0x100 * (x-1), (x = A to B)

Reset value: 0x0000 0000

Writes to this register are ignored if the CRC_BUSY bit is set in the FLASH_xSR register, or CRC_EN bit is set in FLASH_xCR register..

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:2 **CRC_END_ADDR[19:2]**: CRC end address

This register is used when CRC_BY_PAGE is cleared. It must be programmed to the end address of the memory area on which the CRC calculation is performed on bank x.

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

5.9.12 FLASH CRC data register (FLASH_CRCDATAR)

Address offset: 0x03C, 0x13C

Reset value: 0x0000 0000

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

Bits 31:0 **CRC_DATA[31:0]**: CRC result

This bitfield contains the result of the last CRC calculation that occurred on either bank. The value is valid only when CRC calculation completed (CRCENDF is set in FLASH_xISR register). CRC_DATA is cleared when CRC_EN bit is cleared in both FLASH_xCR register, or when application clears CLEAN_CRC bit in either FLASH_xCRCCR register.

5.9.13 FLASH bank x ECC single error fail address (FLASH_xECCSFADDR)

Address offset: 0x040 + 0x100 * (x-1), (x = A to B)

Reset value: 0x0000 0000

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

Bits 31:0 **SEC_FADD[31:0]**: ECC single error correction fail address

When a single ECC error correction occurs during a read operation, the SEC_FADD bitfield contains the system bus address that generated the error.

This register is automatically cleared when SNECCERRF flag that generated the error is cleared.

Note that only the first address that generated an ECC single error correction error is saved in this register.

5.9.14 FLASH bank x ECC double error fail address (FLASH_xECCDFADDR)

Address offset: 0x044 + 0x100 * (x-1), (x = A to B)

Reset value: 0x0000 0000

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

Bits 31:0 **DED_FADD[31:0]**: ECC double error detection fail address

When a double ECC detection occurs during a read operation, the DED_FADD bitfield contains the system bus address that generated the error.

This register is automatically cleared when the DBECCERRF flag that generated the error is cleared.

Note: Only the first address that generated an ECC double error detection error is saved in this register.

5.9.15 FLASH bank x write page protection status register (FLASH_xWRPSR)

Address offset: 0x050 + 0x100 * (x-1), (x = A to B)

Reset value: 0xXXXX XXXX

This read-only register reflects the current values of the corresponding option bits. This register is impacted by the swapping function (see *Section 5.3.14*).

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

Bits 31:0 **WRPG[31:0]**: Write protection for page group n in bank x (n = 0 to 127)
Each bit reflects the write protection status of the corresponding group of four consecutive pages in bank (0: the group is write protected; 1: the group is not write protected).

Bit 0: Group 0, comprising pages 0 to 3
Bit 1: Group 1, comprising pages 4 to 7
Bit N: Group N, comprising pages 4 x N to 4 x N + 3

Note: In STM32H7PxxG devices with 1 Mbyte flash, bits 31:16 are reserved.

5.9.16 FLASH bank x write page protection status register programming (FLASH_xWRPSRP)

Address offset: 0x054 + 0x100 * (x-1), (x = A to B)

Reset value: 0xXXXX XXXX

(The reset value is the same as for FLASH_xWRPSR)

Application must not write to this register when BUSY and IS_OPTCHANGE bits are set in FLASH_xSR. Writes are ignored if HDPL is different from 0 or 1 in the SBS_HDPLSR register.

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

Bits 31:0 **WRPG[31:0]**: Write protection for page group n in bank x programming (n = 0 to 127)
Write to change corresponding bit in FLASH_xWRPSR

Note: In STM32H7PxxG devices with 1 Mbyte flash, bits 31:16 are reserved and must be kept at reset value.

5.9.17 FLASH bank x hide protection status register (FLASH_xHDPSR)

Address offset: 0x058 + 0x100 * (x-1), (x = A to B)

Reset value: 0x0XXX 0XXX

This read-only register reflects the current values of the corresponding option bits. This register is impacted by the swapping function (see *Section 5.3.14*).

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

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HDP_AREA_END[11:0]**: Hide protection user flash area end for bank x

This option sets the end address that contains the last 256-byte block of the hide protection (HDP) area in user flash area of bank x.
If HDP_AREA_END = HDP_AREA_START all the pages in bank are protected.
If HDP_AREA_END < HDP_AREA_START no pages in bank are protected.

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

Bits 11:0 **HDP_AREA_START[11:0]**: Hide protection user flash area start for bank x

This option sets the start address that contains the first 256-byte block of the hide protection (HDP) area in user flash area of bank x.
If HDP_AREA_END = HDP_AREA_START all the pages in bank are protected.
If HDP_AREA_END < HDP_AREA_START no pages in bank are protected.

5.9.18 FLASH bank x hide protection status register programming (FLASH_xHDPSRP)

Address offset: 0x05C + 0x100 * (x-1), (x = A to B)

Reset value: 0x0XXX 0000

(The reset value is the same as for FLASH_xHDPSR.)

Application must not write to this register when BUSY and IS_OPTCHANGE bits are set in FLASH_xSR. Write are ignored if HDPL is different from 0 or 1 in the SBS_HDPLSR register.

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

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HDP_AREA_END[11:0]**: Hide protection user flash area end programming for bank x
Write to change corresponding option byte bits in FLASH_xHDPSR.
If HDP_AREA_END = HDP_AREA_START all the pages in bank are protected.
If HDP_AREA_END < HDP_AREA_START no pages in bank are protected.

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

Bits 11:0 **HDP_AREA_START[11:0]**: Hide protection user flash area start programming for bank x
Write to change corresponding option byte bits in FLASH_xHDPSR.
If HDP_AREA_END = HDP_AREA_START all the pages in bank are protected.
If HDP_AREA_END < HDP_AREA_START no pages in bank are protected.

5.9.19 FLASH bank x HDP extension register (FLASH_xHDPEXTR)

Address offset: 0x060 + 0x100 * (x-1), (x = A to B)

Reset value: 0x00A3 0000

This register is impacted by the swapping function (see *Section 5.3.14*).

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **HDPEXT_ACCDIS[7:0]**: HDP extension area access disable for bank x
Set this bitfield to configure the HDP extension in user flash of bank x. Access to this bitfield depends on its value.
0xA3: HDP extension area is not active and can be configured with HDP_AREA_EXT[11:0]. Any value can be written in this bitfield.
0x5C: HDP extension area is active, and can be increased (not decreased) with HDP_AREA_EXT[11:0]. Writing 0xA3 to this bitfield is ignored.
Others: HDP extension area is active, and writes to this bitfield and HDP_AREA_EXT[11:0] are ignored.

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

Bits 11:0 **HDP_AREA_EXT[11:0]**: HDP extension area end for bank x
This bitfield defines the number of 256-byte pages added to the nonvolatile HDP user flash area of bank x (HDP extension). If the bitfield equals to zero the HDP extension area is not active, regardless of the value in HDPEXT_ACCDIS[7:0].
Accesses to an active HDP extension area is ignored (for writes) or zeroed (for reads) when HDPL is different from 0, 1 or 2 in SBS_HDPLSR register.

More specifically, if the nonvolatile HDP area size is at least two pages, this bitfield is the number of HDP pages added starting at block offset HDP_AREA_END[11:0] +1 included. Otherwise, if the HDP area size is zero, this bitfield is the number of HDP pages added starting at block offset HDP_AREA_END[11:0] included. HDP_AREA_END[11:0] is defined in FLASH_xHDPSR register.

5.9.20 FLASH options key register (FLASH_OPTKEYR)

Address offset: 0x200

Reset value: 0x0000 0000

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

Bits 31:0 **OCUKEY[31:0]**: Options configuration unlock key

Following values must be written to FLASH_OPTKEYR consecutively to unlock FLASH_OPTCR register:

- 1st key = 0x0819 2A3B
- 2nd key = 0x4C5D 6E7F

Reads to this register returns zero. If above sequence is performed twice locks up the corresponding register/bit until the next system reset, and generates a bus error.

5.9.21 FLASH options control register (FLASH_OPTCR)

Address offset: 0x204

Reset value: 0xX000 0001

When the OPTLOCK bit in this register is set writes to all other bits are ignored.

Write access to this register is controlled by the FLASH_OPTKEYR register.

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **OPTERRIE**: Option byte change error interrupt enable bit

This bit controls if an interrupt has to be generated when an error occurs during an option byte change.

0: No interrupt is generated when an error occurs during an option byte change

1: An interrupt is generated when and error occurs during an option byte change.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **OBKERRIE**: Option byte key error interrupt enable bit

This bit controls if an interrupt has to be generated when OBKERRF is set in FLASH_OPTISR.

0: No interrupt is generated when an OBkey error occurs

1: An interrupt is generated when an OBkey error occurs

Bits 26:2 Reserved, must be kept at reset value.

Bit 1 **OPTSTART**: Options update start

Setting this bit triggers an option byte change operation. Write to this bit is ignored when OPTLOCK bit is cleared.

0: No action or option byte change acknowledged

1: Option byte change requested

Refer to *Section 5.4.3: Option byte words modification* for more details.

It is not possible to start an option byte change operation if a CRC calculation is ongoing. Hence setting OPTSTART is ignored when CRC_BUSY is set in FLASH_xSR register.

Bit 0 **OPTLOCK**: Options lock

When this bit is set write to all other bits in this register, and write to OTP words, option bytes and option bytes keys control registers, are ignored.

0: OTP words, FLASH_OPTCR, and FLASH_xSRP registers are unlocked

1: Writes to OTP words, FLASH_xSRP and to other bits than OPTLOCK in FLASH_OPTCR, are ignored

Clearing this bit requires the correct write sequence to FLASH_OPTKEYR register (see this register for details). If a wrong sequence is executed, or the unlock sequence is performed twice, this bit remains locked until next system reset.

During the write access to set LOCK bit from 0 to 1, it is possible to change the other bits of this register.

5.9.22 FLASH options interrupt status register (FLASH_OPTISR)

Address offset: 0x208

Reset value: 0xXXXX XXXX

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **OPTERRF**: Option byte change error flag

When OPTERRF is set, the option byte change operation did not successfully complete. An interrupt is generated when this flag is raised if the OPTERRIE bit of FLASH_OPTCR register is set. Setting OPTERRF of register FLASH_OPTICR clears this bit.

0: No option byte change errors occurred

1: One or more errors occurred during an option byte change operation.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **OBKERRF**: Option byte key error flag

When OBKERRF is set, the last option byte key reading or programming operation did not successfully complete. An interrupt is generated when this flag is raised if the OBKERRIE bit of FLASH_OPTCR register is set. Setting OBKERRF of register FLASH_OPTICR clears this bit.

0: No option byte key reading or programming error occurred

1: A double error detection was found when loading the requested option byte key, or the programming of a nonvirgin key was detected during the last programming operation.

When OBKERRF is set write to KEYSTART bit in FLASH_OBKCR is ignored.

Bits 26:0 Reserved, must be kept at reset value.

5.9.23 FLASH options interrupt clear register (FLASH_OPTICR)

Address offset: 0x20C

Reset value: 0xXXXX XXXX

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **OPTERRF**: Option byte change error flag

Set this bit to clear OPTERRF flag in FLASH_OPTISR register.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **OBKERRF**: Option byte key error flag

Set this bit to clear OBKERRF flag in FLASH_OPTISR register.

Bits 26:0 Reserved, must be kept at reset value.

5.9.24 FLASH OTP lock status register (FLASH_OTPLSR)

Address offset: 0x210

Reset value: 0x0000 XXXX

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTPL[15:0]**: OTP lock n (n = 0 to 15)

Block *n* corresponds to OTP 16-bit word 32 x *n* to 32 x *n* + 31.
OTPL[n] = 1 indicates that all OTP 16-bit words in OTP Block *n* are locked and can no longer be programmed.
OTPL[n] = 0 indicates that all OTP 16-bit words in OTP Block *n* are not locked and can still be modified.

5.9.25 FLASH OTP lock status register programming (FLASH_OTPLSRP)

Address offset: 0x214

Reset value: 0x0000 XXXX (same as for FLASH_OTPLSR)

Application must not write to this register when BUSY and IS_OPTCHANGE bits are set in FLASH_xSR.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OTPL[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTPL[15:0]**: OTP lock n programming (n = 0 to 15)

Write to change corresponding option byte bit in FLASH_OTPLSR.
OTPL bits can be only be set, not cleared.

5.9.26 FLASH nonvolatile status register (FLASH_NVSR)

Address offset: 0x220

Reset value: 0x0000 XXXX

This read-only register reflects the current values of the corresponding option bits. The default value is used when there is an issue with option byte loading.

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

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

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

Bits 7:0 **NVSTATE[7:0]**: Nonvolatile state

This bitfield defines the nonvolatile state of the device:

0xB4: NVOPEN

0x51: NVCLOSED (default)

others: invalid configuration.

5.9.27 FLASH security status register programming (FLASH_NVSRP)

Address offset: 0x224

Reset value: 0x0000 XXXX (same as FLASH_NVSR)

Application must not write to this register when BUSY and IS_OPTCHANGE bits are set in FLASH_xSR. Write are ignored if HDPL is different from 0 or 1 in the SBS_HDPLSR register.

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

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

Bits 7:0 **NVSTATE[7:0]**: Nonvolatile state programming

Write to change corresponding bits in FLASH_NVSR register:

0xB4: NVOPEN

0x51: NVCLOSE

Detailed sequence to grant option byte change from NVCLOSE to NVOPEN is defined in Flash opening of *Section 5.5.3*. When NVSTATE = 0xB4 (resp. 0x51) writing any other value than 0x51 (resp. 0xB4) triggers an option byte change error (OPTERRF).

5.9.28 FLASH RoT status register (FLASH_ROTSR)

Address offset: 0x228

Reset value: 0xXX00 XXXX

This read-only register reflects the current values of the corresponding option bits. Default values are used when there is an issue with option byte loading.

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

Bits 31:24 **BOOT_LOCK[7:0]**: boot lock
0xB4: Change to SWAP_BANK option bit is always ignored (default).
0x6A: Change to SWAP_BANK option bit is allowed.
Others: invalid configuration.

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:8 **DBG_AUTH[7:0]**: Debug authentication method
Every boot RSS uses this information to enforce a specific debug and regression policy when NVSTATE = NVCLOSE in FLASH_NVSR.
0xB4: No debug allowed, no regression (Locked device) (default).
0x8A, 0x6F: No debug allowed, regression with password (Closed device)
Others: invalid configuration.

Bits 7:0 **OEM_PROVD[7:0]**: OEM provisioned device
Every boot RSS uses this information to select the boot image when NVSTATE = NVCLOSE in FLASH_NVSR.
0xB4: Device has been provisioned by the OEM.
0x6A: Device is not provisioned by the OEM (default).
Others: invalid configuration.

5.9.29 FLASH RoT status register programming (FLASH_ROTSRP)

Address offset: 0x22C

Reset value: 0x0000 XXXX (same as FLASH_ROTSR)

Application must not write to this register when BUSY and IS_OPTCHANGE bits are set in FLASH_xSR. Writes are ignored if HDPL is different from 0 in the SBS_HDPLSR register.

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

Bits 31:24 **BOOT_LOCK[7:0]**: Boot lock programming
Write to change corresponding bits in FLASH_ROTSR register.
Writing any other value than 0xB4 or 0x6A is possible, but it triggers an option byte change error (OPTERRF), and no option byte change occurs.

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:8 **DBG_AUTH[7:0]**: Debug authentication method programming
Write to change corresponding bits in FLASH_ROTSR register.
Writing any other value than 0xB4, 0x8A or 0x6F is possible, but it triggers an option byte change error (OPTERRF), and no option byte change occurs.

Bits 7:0 **OEM_PROVD[7:0]**: OEM provisioned device
Write to change corresponding bits in FLASH_ROTSR register.
Writing any other value than 0xB4 or 0x6A is possible, but it triggers an option byte change error (OPTERRF), and no option byte change occurs.

5.9.30 FLASH option byte key control register (FLASH_OBKCR)

Address offset: 0x230

Reset value: 0x0000 0000

Writes are ignored when KEYSTART is set in this register.

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

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **KEYSTART**: Key option start

Set this bit to start the option byte key operation defined by the KEYPROG and KEYERASE bits (refer to those bitfield for details). The embedded flash memory resets KEYSTART when the corresponding operation has been completed.

Application must set either KEYPROG or KEYERASE bit when setting KEYSTART. If application sets both KEYPROG or KEYERASE bits when setting KEYSTART the hardware immediately clears KEYSTART and triggers an option byte key error (OBKERRF).

Bit 14 **KEYPROG**: Key program

Set this bit with KEYSTART to write a new option byte keys. Keep this bit and KEYERASE bit cleared to read the key instead.

0: Read key. Result of the operation is stored in FLASH_OBKDRx registers, if applicable.

1: Write key stored in FLASH_OBKDRx registers.

Bit 13 **KEYERASE**: Key erase

Set this bit with KEYSTART to erase keys based on FLASH_OBKCLR information (see *Section 5.9.31* for details). The embedded flash memory resets KEYERASE when the erase operation has been completed.

0: No action.

1: Erase keys.

Bits 12:10 Reserved, must be kept at reset value.

Bits 9:8 **NEXTKL[1:0]**: Next key level

0x0: OBKINDEX represents the index of the option byte key stored for the hide protection level indicated in SBS_HDPLSR.

0x1: OBKINDEX represents the index of the option byte key stored for the hide protection level indicated in SBS_HDPLSR plus one (for example if HDPL =1 in SBS_HDPLR the key of level 2 is selected).

0x2 or 0x3: reserved

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **OBKINDEX[4:0]**: Option byte key index

This bitfield represents the index of the option byte key in a given hide protection level. Up to 32 keys can be provisioned per hide protection level (0, 1 or 2).

5.9.31 FLASH option byte key clear register (FLASH_OBKCLR)

Address offset: 0x234

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EKEY31	EKEY30	EKEY29	EKEY28	EKEY27	EKEY26	EKEY25	EKEY24	EKEY23	EKEY22	EKEY21	EKEY20	EKEY19	EKEY18	EKEY17	EKEY16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EKEY15	EKEY14	EKEY13	EKEY12	EKEY11	EKEY10	EKEY9	EKEY8	EKEY7	EKEY6	EKEY5	EKEY4	EKEY3	EKEY2	EKEY1	EKEY0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **EKEY{i}**: Erase key {i}

Set this bit to request the erasing of option byte key of index OBKINDEX[4:0] = {i}. Key is also referenced by the HDP level defined in SBS_HDPLSR plus the bitfield NEXTKL[1:0].

The key is actually erased after the next option byte key programming sequence triggered by the setting of KEYERASE and KEYSTART bits. Clearing a key is mandatory when its value must be modified.

After a successful programming this register is cleared.

5.9.32 FLASH option bytes key data register x (FLASH_OBKDRx)

Address offset: 0x240 + 0x4 * x (x = 0 to 3)

Reset value: 0x0000 0000

Writes are ignored until the KEYSTART bit is cleared in FLASH_OBKCR, or when OBKERRF is set in FLASH_OPTISR register.

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

Bits 31:0 **OBKDATA[31:0]**: option byte key data, bits [31+x:0+x]

Data register used in conjunction with FLASH_OBKCR register.

Reading this register (read value once), or incrementing HDPL value in SBS peripheral automatically clears OBKDATA to 0x0.

5.9.33 FLASH option byte word 1 status register (FLASH_OBW1SR)

Address offset: 0x260

Reset value: 0xXXXX XXXX (see *Table 30: Option byte user words organization*)

This register reflects the current values of the corresponding option bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWAP_BANK	Res.	VDDIO_HSLV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG_FZ_STBY	IWDG_FZ_STOP	Res.
r		r											r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	ECC_ON_SRAM	Res.	Res.	NRST_STBY	NRST_STOP	Res.	IWDG_HW	BOR_LEV[1:0]		Res.	Res.
					r			r	r		r	r	r		

Bit 31 **SWAP_BANK**: Bank swapping
This bit is set when a swapping of physical bank is requested on the device. Effect of swapping on registers is summarized on *Table 28* and *Table 29* in *Section 5.3.14*.
By default physical bank 1 is allocated to the bank A.
0: Following a system or power-on-reset both SWAP_BANK bits are cleared in FLASH_xSR registers (see *Section 5.9.5*).
1: Following a system or power-on-reset both SWAP_BANK bits are set in FLASH_xSR registers (see *Section 5.9.5*).

Bit 30 Reserved, must be kept at reset value.

Bit 29 **VDDIO_HSLV**: I/O high-speed at low-voltage
This bit indicates that the product operates below 2.5 V.
0: Product working in the full voltage range, I/O speed optimization at low-voltage disabled
1: Product operating below 2.5 V, I/O speed optimization at low-voltage feature allowed

Bits 28:19 Reserved, must be kept at reset value.

Bit 18 **IWDG_FZ_STBY**: IWDG standby mode freeze
When set the independent watchdog IWDG is frozen in system Standby mode.
0: Independent watchdog frozen in Standby mode
1: Independent watchdog keep running in Standby mode

Bit 17 **IWDG_FZ_STOP**: IWDG stop mode freeze
When set the independent watchdog IWDG is frozen in system Stop mode.
0: Independent watchdog frozen in Stop mode
1: Independent watchdog keep running in Stop mode

Bits 16:11 Reserved, must be kept at reset value.

Bit 10 **ECC_ON_SRAM**: ECC on SRAM
0: ECC on SRAM option disabled (default)
1: ECC on SRAM option enabled

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **NRST_STBY**: Reset on standby mode
0: Independent WDG generates a reset if STANDBY mode is requested
1: Independent WDG does not generate a reset if STANDBY mode is requested (default)

Bit 6 **NRST_STOP**: Reset on stop mode
0: Independent WDG generates a reset if STOP mode is requested
1: Independent WDG does not generate a reset if STOP mode is requested (default)

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IWDG_HW**: Independent watchdog HW Control
1: IWDG watchdog is controlled by software (default)
0: IWDG watchdog is controller by hardware

Bits 3:2 **BOR_LEV[1:0]**: Brownout level
These bits reflects the power level that generates a system reset.
0x0: BOR OFF, POR/PDR reset threshold level is applied (default)
0x1: BOR Level 1, the threshold level is low (around 2.1 V)
0x2: BOR Level 2, the threshold level is medium (around 2.4 V)
0x3: BOR Level 3, the threshold level is high (around 2.7 V)

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

5.9.34 FLASH option byte word 1 status register programming (FLASH_OBW1SRP)

Address offset: 0x264

Reset value: 0xXXXX XXXX (same as for FLASH_OBW1SR)

Application must not write to this register when BUSY and IS_OPTCHANGE bits are set in FLASH_xSR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWAP_BANK	Res.	VDDIO_HSLV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG_FZ_STBY	IWDG_FZ_STOP	Res.
rw		rw											rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	ECC_ON_SRAM	Res.	Res.	NRST_STBY	NRST_STOP	Res.	IWDG_HW	BOR_LEV[1:0]		Res.	Res.
					rw			rw	rw		rw	rw	rw		

Bit 31 **SWAP_BANK**: Bank swapping programming
Write to change corresponding bit in FLASH_OBW1SR register.
Writes to this bit are ignored when BOOT_LOCK is set in FLASH_ROTSR, or when hide protection level in SBS_HDPLSR register is 3.

Bit 30 Reserved, must be kept at reset value.

Bit 29 **VDDIO_HSLV**: I/O high-speed at low-voltage programming
Write to change corresponding bit in FLASH_OBW1SR register.

Bits 28:19 Reserved, must be kept at reset value.

Bit 18 **IWDG_FZ_STBY**: IWDG standby mode freeze programming
Write to change corresponding bit in FLASH_OBW1SR register.

Bit 17 **IWDG_FZ_STOP**: IWDG stop mode freeze
Write to change corresponding bit in FLASH_OBW1SR register.

Bits 16:11 Reserved, must be kept at reset value.

Bit 10 **ECC_ON_SRAM**: ECC on SRAM
Write to change corresponding bit in FLASH_OBW1SR register.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **NRST_STBY**: Reset on standby mode programming
Write to change corresponding bit in FLASH_OBW1SR register.

Bit 6 **NRST_STOP**: Reset on stop mode programming
Write to change corresponding bit in FLASH_OBW1SR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IWDG_HW**: Independent watchdog HW Control
Write to change corresponding bit in FLASH_OBW1SR register.

Bits 3:2 **BOR_LEV[1:0]**: Brownout level
Write to change corresponding bits in FLASH_OBW1SR register.

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

5.9.35 FLASH register map

Table 37. FLASH register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	FLASH_ACCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRHIGHFREQ[1:0]		LATENCY[3:0]			
	Reset value																											0	1	0	0	1	1
0x004	FLASH_AKEYR	CUKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008 - 0x00C	Reserved																																
0x010	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_EN	PG_OTP	Res.	Res.	Res.	PSN[6:0]							START	FW	BER	PER	PG	LOCK
	Reset value															0	0				0	0	0	0	0	0	0	0	0	0	0	0	1
0x014	FLASH_ASR	SWAP_BANK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IS_OPTCHANGE	IS_ERASE	IS_PROGRAM	CRC_BUSY	QW	WBNE	BUSY
	Reset value	0																									0	0	0	0	0	0	0
0x018 - 0x01C	Reserved																																
0x020	FLASH_AIER	Res.	Res.	Res.	CRCRDERRIE	CRCENDIE	DBECCERRIE	SNECCERRIE	RDSERRIE	Res.	Res.	INCERRIE	OBLERRIE	STRBERRIE	PGSERRIE	WRPERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value				0	0	0	0	0			0	0	0	0	0	0																
0x024	FLASH_AISR	Res.	Res.	Res.	CRCRDERRF	CRCENDF	DBECCERRF	SNECCERRF	RDSERRF	Res.	Res.	INCERRF	OBLERRF	STRBERRF	PGSERRF	WRPERRF	EOPF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value				0	0	0	0	0			0	0	0	0	0	0																
0x028	FLASH_AICR	Res.	Res.	Res.	CRCRDERRF	CRCENDF	DBECCERRF	SNECCERRF	RDSERRF	Res.	Res.	INCERRF	OBLERRF	STRBERRF	PGSERRF	WRPERRF	EOPF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value				0	0	0	0	0			0	0	0	0	0	0																
0x02C	Reserved																																

Table 37. FLASH register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x030	FLASH_ACRCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALL_BANK	CRC_BURST[1:0]		Res.	Res.	CLEAN_CRC	START_CRC	Res.	Res.	Res.	Res.	CLEAN_PAGE	ADD_PAGE	CRC_BY_PAGE	Res.	Res.	CRC_PAGE[6:0]						
	Reset value										0	0	1			0	0					0	0	0			0	0	0	0	0	0	0
0x034	FLASH_ ACRCSADDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_START_ADDR[19:2]																		Res.	Res.
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x038	FLASH_ ACRCEADDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_END_ADDR[19:2]																		Res.	Res.
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x03C	FLASH_ CRCDATAR	CRC_DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	FLASH_ AECCSFADDR	SEC_FADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	FLASH_ AECCDFADDR	DED_FADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048 - 0x04C	Reserved																																
0x050	FLASH_AWRPSR	WRPG[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x054	FLASH_ AWRPSRP	WRPG[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x058	FLASH_AHDPSR	Res.	Res.	Res.	Res.	HDP_AREA_END[11:0]												Res.	Res.	Res.	Res.	HDP_AREA_START[11:0]											
	Reset value					X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x05C	FLASH_AHDPSRP	Res.	Res.	Res.	Res.	HDP_AREA_END[11:0]												Res.	Res.	Res.	Res.	HDP_AREA_START[11:0]											
	Reset value					X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x060	FLASH_ AHDPEXTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPEXT_ACCDIS [7:0]								Res.	Res.	Res.	Res.	HDP_AREA_EXT[11:0]											
	Reset value									1	0	1	0	0	0	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x064 - 0x0FC	Reserved																																

Table 37. FLASH register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x100	FLASH_ACCR_ALIAS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRHIGHFREQ[1:0]		LATENCY[3:0]			
	Reset value																											0	1	0	0	1	1
0x104	FLASH_BKEYR	CUKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108 - 0x10C	Reserved																																
0x110	FLASH_BCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_EN	PG_OTP	Res.	Res.	Res.	PSN[6:0]							START	FW	BER	PER	PG	LOCK
	Reset value															0	0				0	0	0	0	0	0	0	0	0	0	0	0	1
0x114	FLASH_BSR	SWAP_BANK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IS_OPTCHANGE	IS_ERASE	IS_PROGRAM	CRC_BUSY	QW	WBNE	BUSY
	Reset value	0																									0	0	0	0	0	0	0
0x118 - 0x11C	Reserved																																
0x120	FLASH_BIER	Res.	Res.	Res.	CRCRDERRIE	CRCENDIE	DBECCERRIE	SNECCERRIE	RDSERRIE	Res.	Res.	INCERRIE	OBLERRIE	STRBERRIE	PGSERRIE	WRPERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value				0	0	0	0	0			0	0	0	0	0	0																
0x124	FLASH_BISR	Res.	Res.	Res.	CRCRDERRF	CRCENDF	DBECCERRF	SNECCERRF	RDSERRF	Res.	Res.	INCERRF	OBLERRF	STRBERRF	PGSERRF	WRPERRF	EOPF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value				0	0	0	0	0			0	0	0	0	0	0																
0x128	FLASH_BICR	Res.	Res.	Res.	CRCRDERRF	CRCENDF	DBECCERRF	SNECCERRF	RDSERRF	Res.	Res.	INCERRF	OBLERRF	STRBERRF	PGSERRF	WRPERRF	EOPF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value				0	0	0	0	0			0	0	0	0	0	0																
0x12C	Res.																																

Table 37. FLASH register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x130	FLASH_BCRCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALL_BANK	CRC_BURST[1:0]		Res.	Res.	CLEAN_CRC	START_CRC	Res.	Res.	Res.	Res.	CLEAN_PAGE	ADD_PAGE	CRC_BY_PAGE	Res.	Res.	CRC_PAGE[6:0]						
	Reset value										0	0	1			0	0					0	0	0			0	0	0	0	0	0	0
0x134	FLASH_ BCRCSADDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_START_ADDR[19:2]																		Res.	Res.
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x138	FLASH_ BCRCEADDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRC_END_ADDR[19:2]																		Res.	Res.
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x13C	FLASH_ CRCDATAR_ ALIAS	CRC_DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x140	FLASH_ BECCSFADDR	SEC_FADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x144	FLASH_ BECCDFADDR	DED_FADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x148 - 0x14C	Reserved																																
0x150	FLASH_BWRPSR	WRPG[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x154	FLASH_ BWRPSRP	WRPG[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x158	FLASH_BHDPSR	Res.	Res.	Res.	Res.	HDP_AREA_END[11:0]												Res.	Res.	Res.	Res.	HDP_AREA_START[11:0]											
	Reset value					X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x15C	FLASH_BHDPSRP	Res.	Res.	Res.	Res.	HDP_AREA_END[11:0]												Res.	Res.	Res.	Res.	HDP_AREA_START[11:0]											
	Reset value					X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x160	FLASH_ BHDPEXTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPEXT_ACCDIS [7:0]								Res.	Res.	Res.	Res.	HDP_AREA_EXT[11:0]											
	Reset value									1	0	1	0	0	0	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x164 - 0x1FC	Reserved																																
0x200	FLASH_OPTKEYR	OCUKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37. FLASH register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x204	FLASH_OPTCR	Res.	OPTERRIE	Res.	Res.	OBKERRIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTSTART	OPTLOCK
	Reset value		0			0																										0	1
0x208	FLASH_OPTISR	Res.	OPTERRF	Res.	Res.	OBKERRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0			0																											
0x20C	FLASH_OPTICR	Res.	OPTERRF	Res.	Res.	OBKERRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0			0																											
0x210	FLASH_OTPLSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTPL[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x214	FLASH_OTPLSRP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTPL[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x218 - 0x21C	Reserved																																
0x220	FLASH_NVSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NVSTATE[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x224	FLASH_NVSRP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NVSTATE[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x228	FLASH_ROTSR	BOOT_LOCK[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_AUTH[7:0]								OEM_PROVD[7:0]							
	Reset value	X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x22C	FLASH_ROTSRP	BOOT_LOCK[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_AUTH[7:0]								OEM_PROVD[7:0]							
	Reset value	X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x230	FLASH_OBKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYSTART	KEYPROG	KEYERASE	Res.	Res.	Res.	NEXTKL[1:0]		Res.	Res.	Res.	OBKINDEX[4:0]				
	Reset value																	0	0	0				0	0				0	0	0	0	0
0x234	FLASH_OBKCLR	EKEY31	EKEY30	EKEY29	EKEY28	EKEY27	EKEY26	EKEY25	EKEY24	EKEY23	EKEY22	EKEY21	EKEY20	EKEY19	EKEY18	EKEY17	EKEY16	EKEY15	EKEY14	EKEY13	EKEY12	EKEY11	EKEY10	EKEY9	EKEY8	EKEY7	EKEY6	EKEY5	EKEY4	EKEY3	EKEY2	EKEY1	EKEY0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37. FLASH register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x238 - 0x23C	Reserved																																
0x240 to 0x24C	FLASH_OBKDR0 to FLASH_OBKDR3	OBKDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x250 - 0x254	Reserved																																
0x258	FLASH_FEATSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLASH_SIZE[2:0]			CNN_HSP_DIS	CAN_DIS	CRYPT_DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											X	X	X	X	X	X																
0x25C	FLASH_FEATSRP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLASH_SIZE[2:0]			CNN_HSP_DIS	CAN_DIS	CRYPT_DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											X	X	X	X	X	X																
0x260	FLASH_OBW1SR	SWAP_BANK	Res.	VDDIO_HSLV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG_FZ_STBY	IWDG_FZ_STO	Res.	Res.	Res.	Res.	Res.	Res.	ECC_ON_SRAM	Res.	Res.	NRST_STBY	NRST_STOP	Res.	IWDG_HW	BOR_LEV[1:0]		Res.	Res.
	Reset value	X		X											X	X							X			X	X		X	X	X		
0x264	FLASH_ OBW1SRP	SWAP_BANK	Res.	VDDIO_HSLV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG_FZ_STBY	IWDG_FZ_STO	Res.	Res.	Res.	Res.	Res.	Res.	ECC_ON_SRAM	Res.	Res.	NRST_STBY	NRST_STOP	Res.	IWDG_HW	BOR_LEV[1:0]		Res.	Res.
	Reset value	X		X											X	X							X			X	X		X	X	X		

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

6 Power control (PWR)

6.1 Introduction

The power control section (PWR) provides an overview of the supply architecture for the different power domains and of the supply configuration controller.

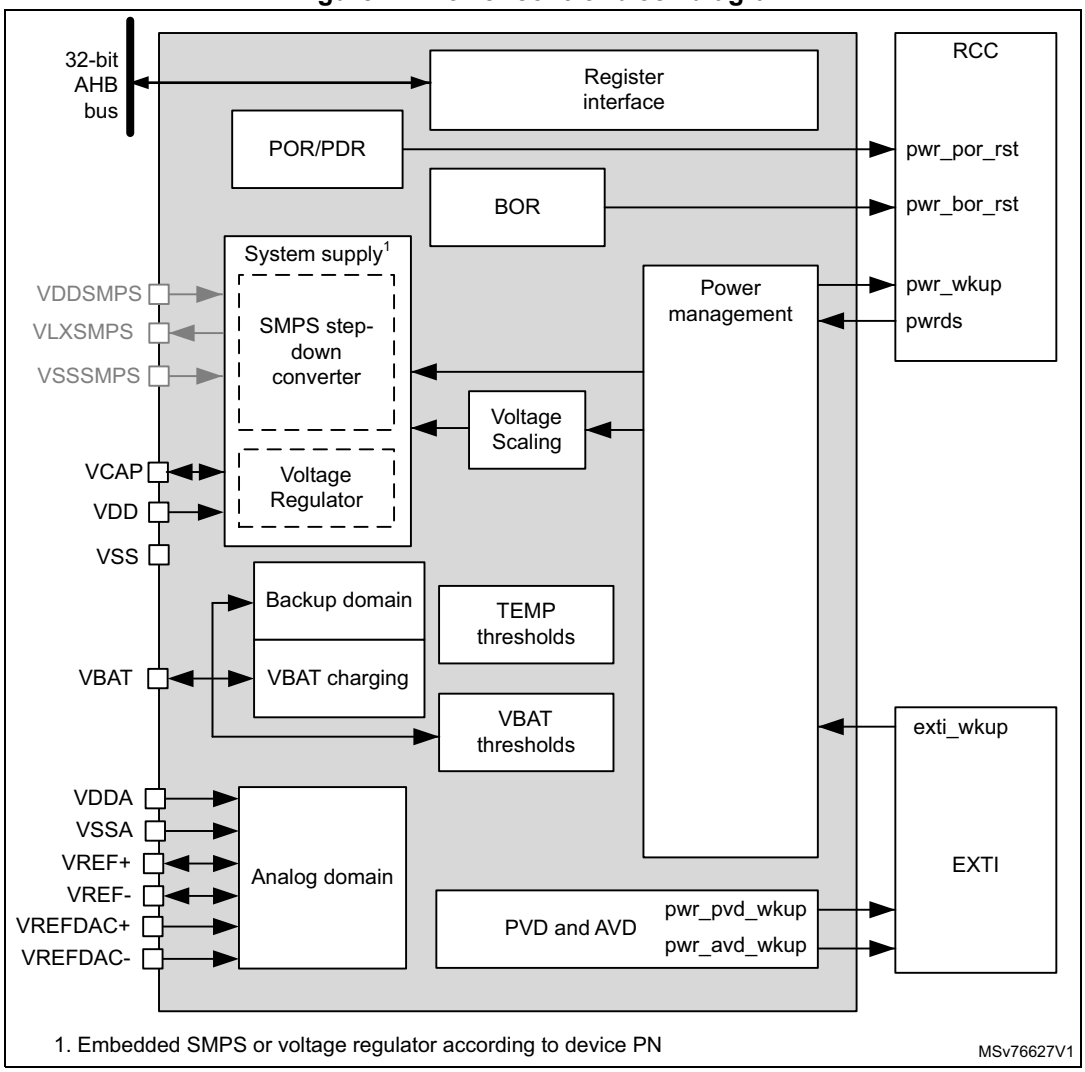
It also describes the features of the power supply supervisors and explains how the V_{CORE} supply domain is configured depending on the operating modes, the selected performance (clock frequency) and the voltage scaling.

6.2 PWR main features

- Power supplies and supply domains
  - Core domains (V_{CORE})
  - V_{DD} domain
  - Backup domain (V_{SW}, V_{BKP})
  - Analog domain (V_{DDA})
- System supply voltage regulation
  - Integrated switched-mode power supply power-efficient voltage down-converter (SMPS step-down converter) or voltage regulator (LDO)
- Power supply supervision
  - POR/PDR monitor
  - BOR monitor
  - PVD monitor
  - AVD monitor
  - V_{BAT} thresholds
  - Temperature thresholds
- Power management
  - V_{BAT} battery charging
  - Operating modes
  - Voltage scaling control
  - Low-power modes

6.3 PWR block diagram

Figure 17. Power control block diagram



6.3.1 PWR pins and internal signals

Table 38 lists the PWR inputs and output signals connected to package pins or balls, while *Table 39* shows the internal PWR signals.

Table 38. PWR input/output signals connected to package pins or balls

Pin name	Signal type	Description
VDD	Supply input	Main I/O and V_{DD} domain supply input
VDDA	Supply input	External analog power supply for analog peripherals
VSSA	Supply	Analog peripherals ground
VREFDAC+, VREFDAC-	Supply input	External reference voltage for DACs

Table 38. PWR input/output signals connected to package pins or balls (continued)

Pin name	Signal type	Description
VREF+,VREF-	Supply input/output	External reference voltage for ADCs
VBAT	Supply input/output	Backup battery supply input
VDDSMPS	Supply input	Step-down converter supply input(1)
VLXSMPS	Supply output	Step-down converter supply output(1)
VSSSMPS	Supply input	Step-down converter ground(1)
VCAP	Supply Input/Outputs	Digital core domain supply
VSS	Supply	Main ground
WKUPx	Input	Wakeup pins

1. If embedded SMPS option

Table 39. PWR internal input/output signals

Signal name	Signal type	Description
AHB	Input/output	AHB register interface
pwr_pvd_wkup	Output	Programmable voltage detector output
pwr_avd_wkup	Output	Analog voltage detector output
pwr_por_rst	Output	Power-on reset
pwr_bor_rst	Output	Brownout reset
exti_wkup	Input	Wakeup request
pwr_wkup	Output	Bus matrix clock wakeup request

6.4 Power supplies

The device uses the following power supply pins:

- V_{DD}: external power supply for I/Os and system analog blocks such as reset, power management and oscillators, as well as the backup domain if no battery is present. VDD also supplies the integrated voltage regulator when enabled.
- V_{BAT}: optional external power supply for backup domain when V_{DD} is not present (V_{BAT} mode).

  This power supply must be connected to VDD when no battery is used.
- V_{DDSMPS} (SMPS option): external power supply for the SMPS step-down converter.

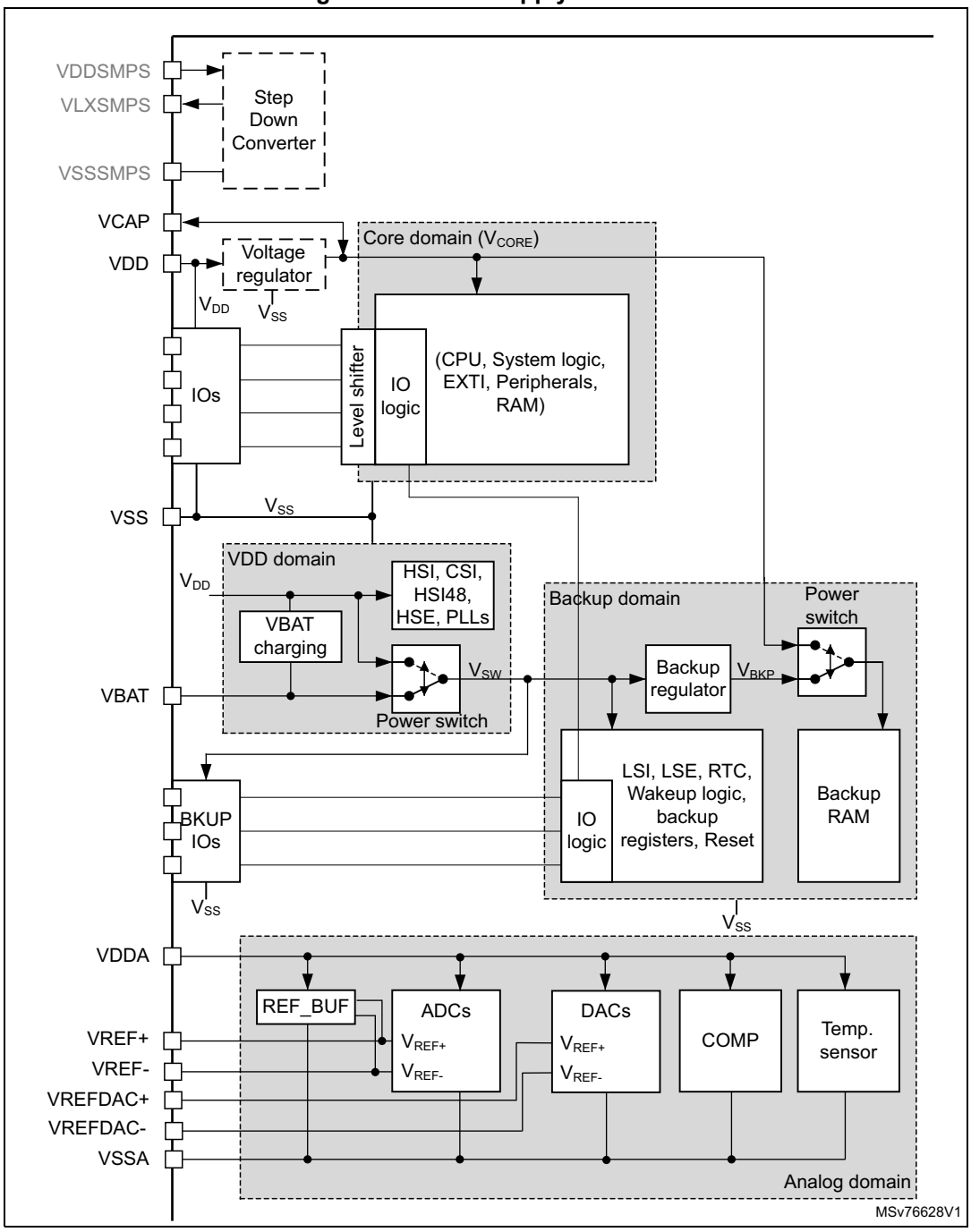
  VDDSMPS must be connected to VDD or tied to VSS when the SMPS is not used.
- V_{LXSMPS} (SMPS option): step-down converter supply output.
- V_{SSSMPS} (SMPS option): separate step-down converter ground.
- V_{CAP}: digital core domain (V_{CORE}) supply

  This power supply is independent from all the other power supplies:
  - (LDO option) When the embedded voltage regulator is enabled, V_{CORE} is delivered by the internal voltage regulator. An external decoupling capacitor can be connected to the VCAP pin.
  - (SMPS option) V_{CORE} is delivered from the SMPS step-down converter through the VCAP pin.
  - (LDO or SMPS bypass) V_{CORE} is delivered from an external power supply through the VCAP pin.
- V_{DDA}: external analog power supply for ADCs and voltage reference buffers.

  This power supply is independent from all the other power supplies.
- V_{REF+}: external reference voltage for ADC.
  - When the voltage reference buffer is enabled, V_{REF+} and V_{REF-} are delivered by the internal voltage reference buffer.
  - When the voltage reference buffer is disabled, V_{REF+} is delivered by an independent external reference supply.
- V_{SSA}: separate analog and reference voltage ground.
- V_{SS}: common ground for all supplies except for step-down converter and analog peripherals.

Note: Depending on the operating power supply range, some peripherals might be used with limited features and performance. For more details, refer to section "General operating conditions" of the device datasheets.

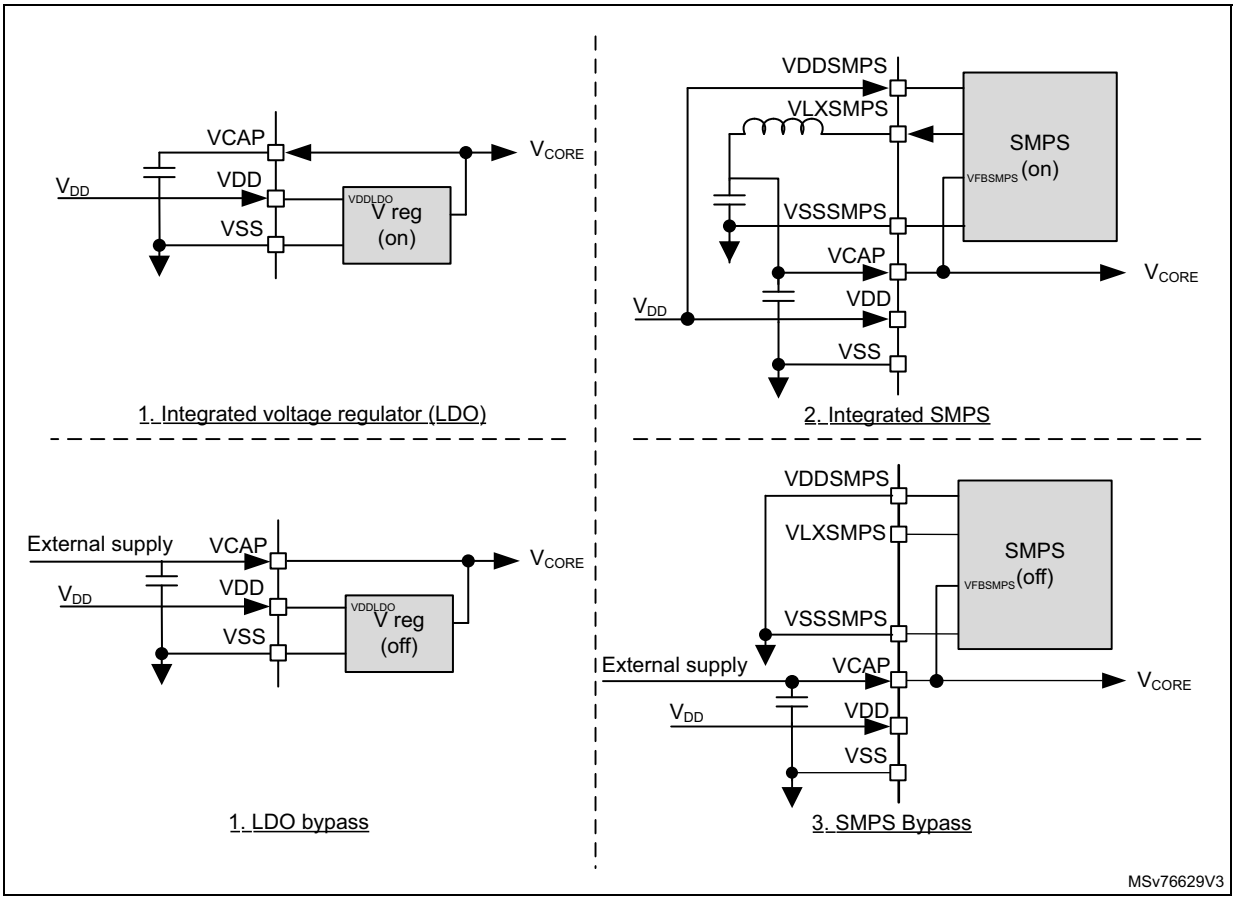
Figure 18. Power supply overview



Depending on the selected product option there are four possible configurations for supplying V_{CORE} illustrated in *Figure 19*.

1. (LDO option) Integrated voltage regulator is used to supply the core voltage (low cost solution in terms of external components).
2. (LDO option) Integrated voltage regulator not used (bypassed). The core voltage comes from an external source.
3. (SMPS option) Integrated SMPS step down converter is used to directly supply the core voltage (high efficiency solution).
4. (SMPS option) Integrated SMPS not used (bypassed). The core voltage comes from an external source.

Figure 19. Supply configurations[a]



6.4.1 System supply startup

The system startup sequence from power-on in different supply configurations is the following (see *Figure 20*):

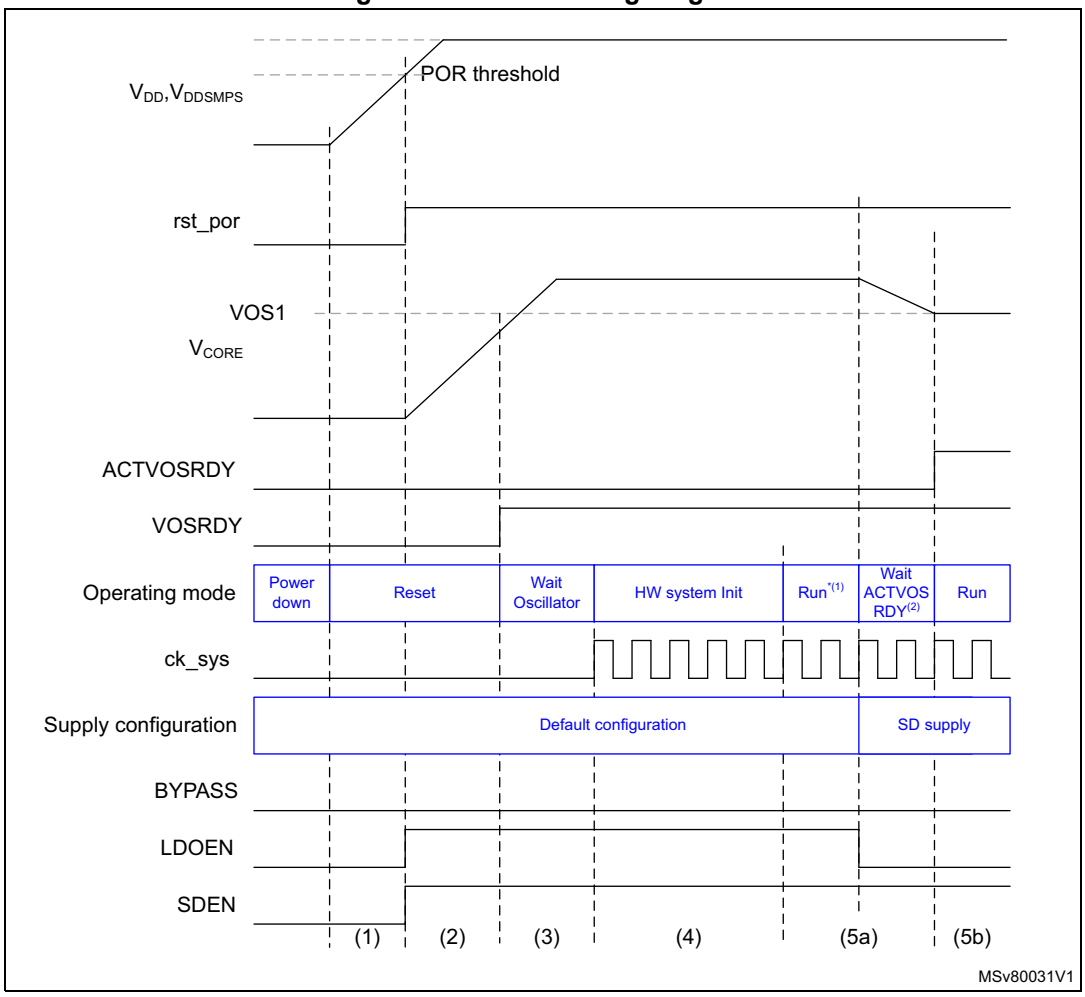
a. Refer to application note ANxxxx (TBD) “Getting started hardware with STM32H7Pxxx MCUs" for the possible power scheme and connected capacitors.

1. When the system is powered on, the POR monitors V_{DD} supply. Once V_{DD} is above the POR threshold level, the integrated SMPS step-down converter or voltage regulator is enabled in the default supply configuration.
2. The system is kept in reset as long as V_{CORE} is not stable.
3. Once V_{CORE} is stable, the system is taken out of reset and the HSI oscillator is enabled.
4. Once the oscillator is stable, the system is initialized: flash memory is ready, option bytes are loaded and the CPU starts in limited run mode (Run*).
5. The software must program the BYPASS bit in the PWR control status register 2 (PWR_CSR2) to indicate whether an external power supply is connected (bypass mode). If an external power supply is connected, the internal power supply (LDO or SMPS) is disabled.
   The software must then poll the ACTVOSRDY bit in the *PWR status register 1 (PWR_SR1)* to guarantee valid voltage levels:
   a) As long as ACTVOSRDY indicates that voltage levels are invalid, the system is in Run* mode, write accesses to the RAMs are not permitted and VOS must not be changed.
   b) Once ACTVOSRDY indicates that voltage levels are valid, the system is in normal Run mode, write accesses to RAMs are allowed and VOS can be changed.

V_{CORE} supplied from the integrated voltage regulator or SMPS step-down converter

When V_{CORE} is supplied from the integrated voltage regulator or SMPS step-down converter, the V_{CORE} voltage first settles at V_{CORE} default level (VOS1).

Figure 20. Device startup with V_{CORE} supplied from integrated SMPS or voltage regulator



1. In Run* mode, write operations to RAM are not allowed.
2. Write operations to RAM are allowed and VOS can be changed only when ACTVOSRDY is valid.

When exiting from Standby mode, the software must wait for the ACTVOSRDY bit to be set in *PWR status register 1 (PWR_SR1)* to indicate V_{CORE} voltage levels are valid, before performing write accesses to RAM or changing VOS.

V_{CORE} supplied in Bypass mode (internal regulator OFF)

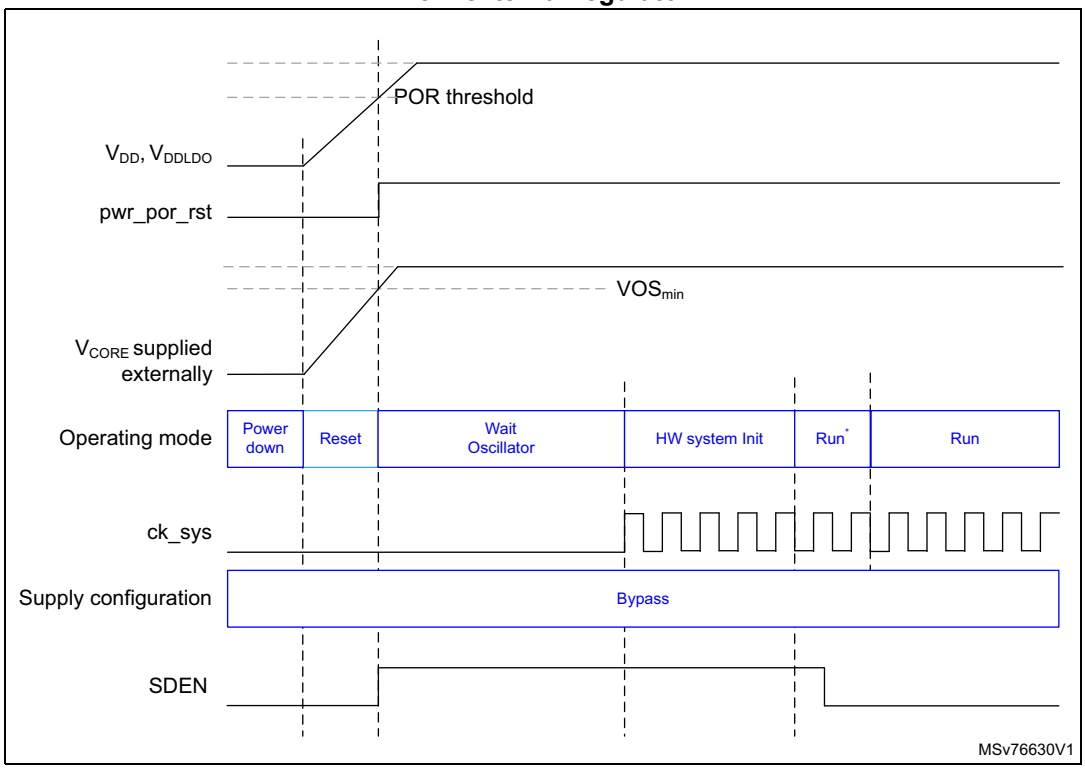
The external V_{CORE} voltage can be adjusted according to the user application needs (refer to section *General operating conditions* of the datasheet for details on V_{CORE} level versus the maximum operating frequency).

Caution: When supplying V_{CORE} from an external supply, care must be taken to respect the start-up sequence shown in *Figure 21*. In particular failure to respect the following constraints may cause damage to the device:

- $V_{CORE} \leq V_{DD}$ at all times.
- $V_{CORE} > VOS_{min}$ when $V_{DD} >$ POR threshold

Once the device has entered Run* mode, the software must disable the internal supply by setting BYPASS = 0 in *PWR control status register 2 (PWR_CSR2)*.

Figure 21. Device startup sequence with V_{CORE} supplied in Bypass mode from external regulator



6.4.2 Core domain

The V_{CORE} core domain supply can be provided by the integrated SMPS step-down converter or LDO voltage regulator, depending on the part number, or by an external supply (VCAP). V_{CORE} supplies all the digital circuitries except for the backup domain and the Standby circuitry.

LDO voltage regulator

The embedded voltage regulator (LDO) requires external capacitors to be connected to VCAP pins.

The voltage regulator provides three different operating modes: high power (HP), low power (LP) or Off. These modes are used depending on the system operating modes (Run, Stop and Standby).

- Run modes

  The LDO regulator operates in high-power mode and delivers full power to the V_{CORE} domain (core, memories and digital peripherals). The regulator output voltage can be scaled by software to different voltage levels (VOS0, VOS1 and VOS2) that are configured through the VOS bitfield in the *PWR control status register 4 (PWR_CSR4)*. The VOS voltage scaling allows optimization of the power consumption when the system is clocked below the maximum frequency. By default VOS1 is selected after system reset. VOS can be changed on-the-fly to adapt to the required system performance.

- Stop mode

  The LDO regulator operates in LP mode and supplies the V_{CORE} domain to retain the content of registers and internal memories. In LP mode only SVOS low and SVOS high voltage levels are allowed. The voltage level is selected through the SVOS bit in the *PWR control register 1 (PWR_CR1)*, depending on whether peripheral clocks are active or not in STOP mode.

- Standby mode

  The voltage regulator is OFF and the V_{CORE} domains are powered down. The content of the registers and memories is lost except for the Standby circuitry and the backup domain.

For more details, refer to the voltage regulator section in the datasheets.

SMPS step-down converter regulator

The SMPS step-down converter requires:

- An external coil to connect between the dedicated VLXSMPS pin and VCAP
- An external capacitor to connect between VCAP and VSSSMPS

The SMPS step-down converter works in three different power modes: high power (HP), low power (LP) or Off.

When the SMPS step-down converter is used in internal supply mode, the converter operating modes depend on the system modes (Run, Stop, Standby) and are configured through the associated VOS and SVOS levels:

- Run mode

  The SMPS step-down converter operates in HP mode and provides full power to the V_{CORE} domain (core, memories and digital peripherals). The regulator output voltage can be scaled by software to different voltage levels (VOS0, VOS1 and VOS2) that are configured through the VOS bitfield in the *PWR control status register 4 (PWR_CSR4)*. The VOS voltage scaling allows optimization of the power consumption when the system is clocked below the maximum frequency. By default VOS1 is selected after system reset. VOS can be changed on-the-fly to adapt to the required system performance.

- Stop mode

  The SMPS step-down converter operates in LP mode and supplies the V_{CORE} domain to retain the content of registers and internal memories. In LP mode only SVOS low and SVOS high voltage levels are allowed. The voltage level is selected through the SVOS bit in the *PWR control register 1 (PWR_CR1)*, depending on whether peripheral clocks are active or not in STOP mode.

- Standby mode

  The SMPS step-down converter is OFF and the V_{CORE} domains are powered down. The content of the registers and memories are lost except for the Standby circuitry and the backup domain.

External supply

When V_{CORE} is supplied from an external source (Bypass mode), different operating modes can be used depending on the system operating modes (Run, Stop or Standby):

- In Run modes

  The external source supplies full power to the V_{CORE} domain (core, memories and digital peripherals). The external source output voltage is scalable through different voltage levels (VOS0 and VOS1 - VOS2 is not available in Bypass mode). The externally applied voltage level must be reflected in the VOS bit of *PWR control status register 4 (PWR_CSR4)* register. The RAMs must only be accessed for write operations when the external applied voltage level matches VOS settings.

- In Stop mode

  The external V_{CORE} supply should be maintained higher than VOS_{min} to ensure proper internal wakeup.

- In Standby mode

  The wake mechanism should be monitored externally. Refer to the datasheet for the proper V_{CORE} ramp time.

6.4.3 Backup domain

To retain the content of the backup domain (RTC, backup registers and backup RAM) when V_{DD} is turned off, VBAT pin can be connected to an optional voltage which is supplied from a battery or from another source.

The switching to V_{BAT} is controlled by the power-down reset embedded in the Reset block that monitors the V_{DD} supply.

Warning: **During $t_{RSTTEMPO}$ (time delay at V_{DD} startup) or after a PDR is detected, the power switch between V_{BAT} and V_{DD} remains connected to V_{BAT}.**
During the startup phase, if V_{DD} is established in less than $t_{RSTTEMPO}$ (see the datasheet for the value of $t_{RSTTEMPO}$) and $V_{DD} > V_{BAT} + 0.6$ V, a current may be injected into V_{BAT} through an internal diode connected between V_{DD} and the power switch (V_{BAT}).
If the power supply/battery connected to the VBAT pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the VBAT pin.

When the V_{DD} supply is present, the backup domain is supplied from V_{DD}. This allows saving V_{BAT} power supply battery life time.

If no external battery is used in the application, it is recommended to connect VBAT externally to VDD, and add a 100 nF external ceramic capacitor between VBAT and VSS.

When the V_{DD} supply is present and higher than the PDR threshold, the backup domain is supplied by V_{DD} and the following functions are available:

- PC14 and PC15 can be used either as GPIO or as LSE pins.
- PC13 can be used either as GPIO or as RTC_OUT1, RTC_TS, TAMP_IN1, TAMP_OUT2 pin assuming they have been configured by the RTC or the TAMPER registers (see *Section 42.3.3: GPIOs controlled by the RTC*).

Note: Since the switch only sinks a limited amount of current, the use of PC13, PC14 and PC15 GPIOs is restricted: only one I/O can be used as an output at a time. These I/Os must not be used as current sources (for example to drive an LED). Refer to the product datasheet.

In V_{BAT} mode, when the V_{DD} supply is absent and a supply is present on V_{BAT}, the backup domain is supplied by V_{BAT} and the following functions are available:

- PC14 and PC15 can be used as LSE pins only.
- PC13 can be used as RTC_OUT1, RTC_TS, TAMP_IN1, TAMP_OUT2 pin assuming they have been configured by the RTC or the TAMPER registers (see *Section 42.3.3: GPIOs controlled by the RTC*).

Accessing the backup domain

After reset, the backup domain (RTC registers and RTC backup registers) is protected against possible unwanted write accesses. To enable access to the backup domain, set the DBP bit in the *PWR control register 1 (PWR_CR1)*.

For more detail on RTC and backup RAM access, refer to *Section 7: Reset and clock control (RCC)*.

Backup RAM

The backup domain includes 4 Kbytes of backup RAM accessible in 32-bit, 16-bit or 8-bit data mode. The backup RAM is supplied from the backup regulator in the backup domain. When the backup regulator is enabled through BREN bit in the *PWR control status register 1 (PWR_CSR1)*, the backup RAM content is retained even in Standby and/or V_{BAT} mode (it can be considered as an internal EEPROM if V_{BAT} is always present.)

The backup regulator can be ON or OFF depending whether the application needs the backup RAM function in Standby or V_{BAT} modes.

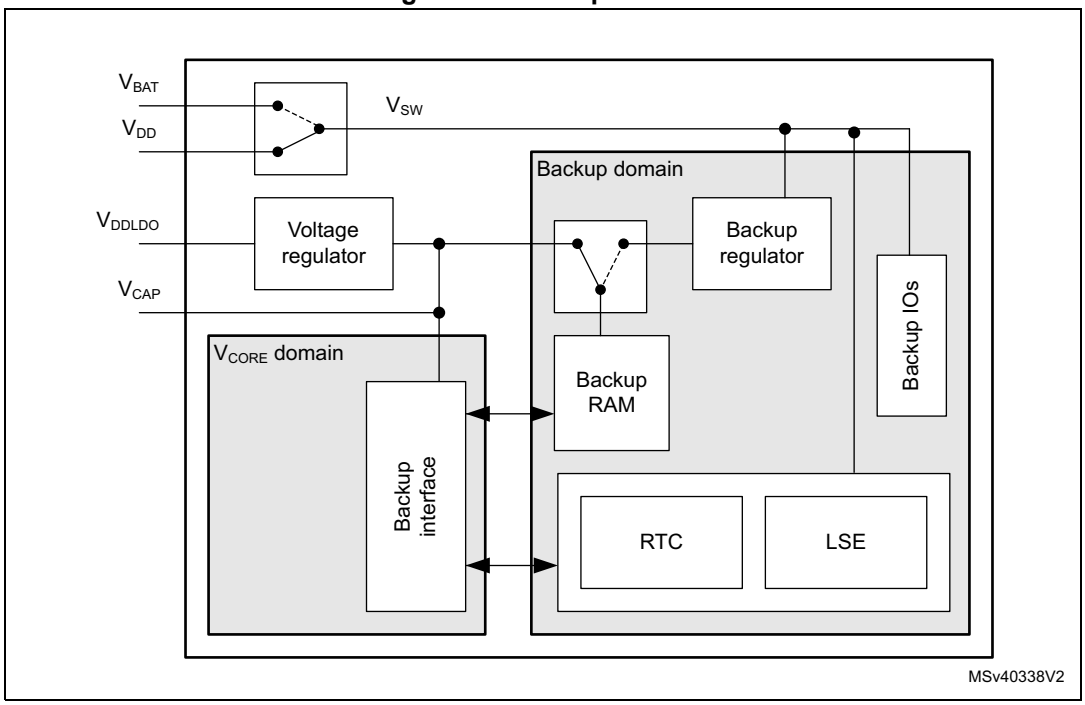
After a tamper event, if the RTC flag is set, the backup RAM cannot be used until an erase is explicitly requested. When a tamper event is generated, reading the backup RAM returns 0x0, and any write different from 0 is not effective. The backup RAM can be erased:

- Through the flash memory interface when a regression from nvstate closed to nvstate open is requested. See also *Section 4: System security (TBD)*.

  or

- By performing a simple dummy write with zero as data to any address of the backup RAM (this action resets the whole backup RAM when it is performed after a tamper event).

Figure 22. Backup domain



6.4.4 V_{BAT} battery charging

When V_{DD} is present, the external battery connected to VBAT can be charged through an internal resistance.

V_{BAT} charging can be performed either through a 5 kΩ resistor or through a 1.5 kΩ resistor, depending on the VBRS bit value in *PWR control status register 2 (PWR_CSR2)*.

The battery charging is enabled by setting the VBE bit in the *PWR control status register 2 (PWR_CSR2)*. It is automatically disabled in V_{BAT} mode.

6.4.5 Analog supply

Separate V_{DDA} analog supply

The analog domain is supplied by dedicated V_{DDA} and V_{SSA} pins that allow the supply to be filtered and shielded from noise on the PCB, thus improving ADC conversion accuracy:

- The analog supply voltage input is on a separate VDDA pin.
- An isolated supply ground connection is provided on the VSSA pin.

Analog reference voltage V_{REF+}/V_{REF-}

V_{REF+} defines the upper limit of the ADC input voltage, represented by the full-scale value. V_{REF-} defines the lower limit, represented by the value 0.

When enabled by ENVR bit in the VREFBUF control and status register (see *Section 28: Voltage reference buffer (VREFBUF)*), V_{REF+} is provided by the internal voltage reference buffer and output on the VREF+ pin. V_{REF-} (connected to V_{SSA}) is output on VREF-.

When the internal voltage reference buffer is disabled by ENVR, the voltage reference can be provided by an external reference connected between the VREF+ and VREF- pins.

*Note: VREF+ and VREF- pins are not available on all packages (in this case they are connected internally respectively to VDDA and VSSA). **Do not** enable the internal voltage reference buffer when an external reference is connected.*

DAC reference voltage $DACV_{REF+}$/$DACV_{REF-}$

The DACs have a dedicated reference voltage. $DACV_{REF+}$ defines the upper limit (full scale) DAC output voltage and $DACV_{REF-}$ defines the lower limit (0 value). The DAC reference is supplied from an external reference connected between the DACVREF+ and DACVREF- pins. If the internal voltage reference buffer (VREFBUF) is enabled, it can be used to supply the DAC reference by connecting VREF+ to DACVREF+ and VREF- to DACVREF- externally. In this case, the DAC and ADC clocks should be synchronized, to avoid polluting the ADC with noise injected by the DAC, or vice versa.

6.5 Power supply supervision

Power supply level monitoring is available on the following supplies:

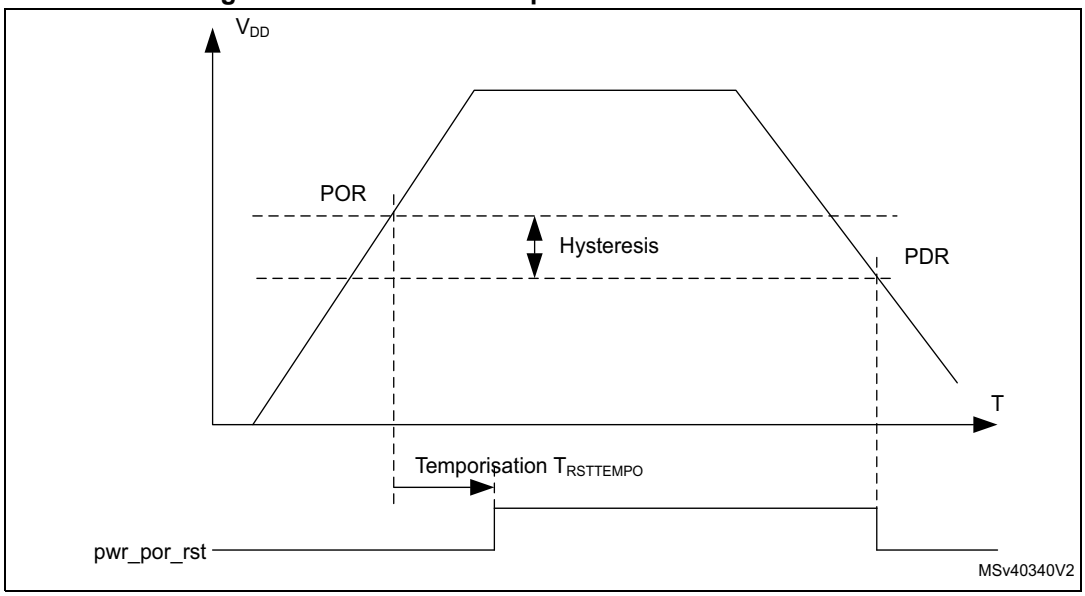
- V_{DD} (V_{DDSMPS}) can be monitored via POR/PDR (see *Section 6.5.1*), BOR (see *Section 6.5.2*) and PVD monitor (see *Section 6.5.3*)
- V_{DDA} can be monitored via AVD monitor (see *Section 6.5.4*)
- V_{BAT} can be monitored via V_{BAT} threshold (see *Section 6.5.5*)
- V_{SW} can be monitored via rst_vsw, which keeps the V_{SW} domain in Reset mode as long as the level is not OK.
- V_{BKP} can be monitored via a BRRDY bit in the *PWR control status register 1 (PWR_CSR1)*.
- V_{FBSMPS} (V_{CORE}) can be monitored via an SDRDY bit in the *PWR control status register 2 (PWR_CSR2)*.

6.5.1 Power-on reset (POR)/power-down reset (PDR)

The system has an integrated POR/PDR circuitry that ensures proper startup operation.

The system remains in Reset mode when V_{DD} is below a specified V_{POR} threshold, without the need for an external reset circuit. Once the V_{DD} supply level is above the V_{POR} threshold, the system is taken out of reset (see *Figure 23*). For more details concerning the power-on/power-down reset threshold, refer to the electrical characteristics section of the datasheets.

Figure 23. Power-on reset/power-down reset waveform



1. For thresholds and hysteresis values, refer to the datasheets.

6.5.2 Brownout reset (BOR)

During power-on, the Brownout reset (BOR) keeps the system under reset until the V_{DD} supply voltage reaches the specified V_{BOR} threshold.

The V_{BOR} threshold is configured through system option bytes. By default, BOR is OFF. The following programmable V_{BOR} thresholds can be selected:

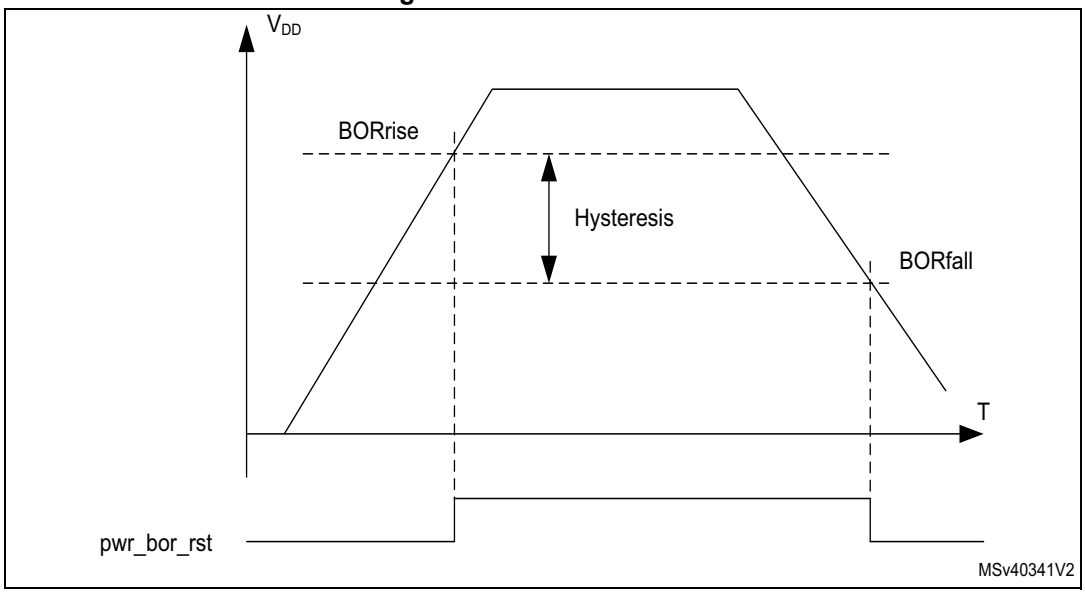
- BOR OFF (V_{BOR0})
- BOR Level 1 (V_{BOR1})
- BOR Level 2 (V_{BOR2})
- BOR Level 3 (V_{BOR3})

For more details on the brown-out reset thresholds, refer to the section “Electrical characteristics” of the product datasheet.

A system reset is generated when the BOR is enabled and the V_{DD} supply voltage drops below the selected V_{BOR} threshold.

BOR can be disabled by programming the system option bytes. To disable the BOR function, V_{DD} must have been higher than V_{BOR0} to start the system option byte programming sequence. The power-down is then monitored by the PDR (see *Section 6.5.1*).

Figure 24. BOR thresholds



1. For thresholds and hysteresis values, refer to the datasheets.

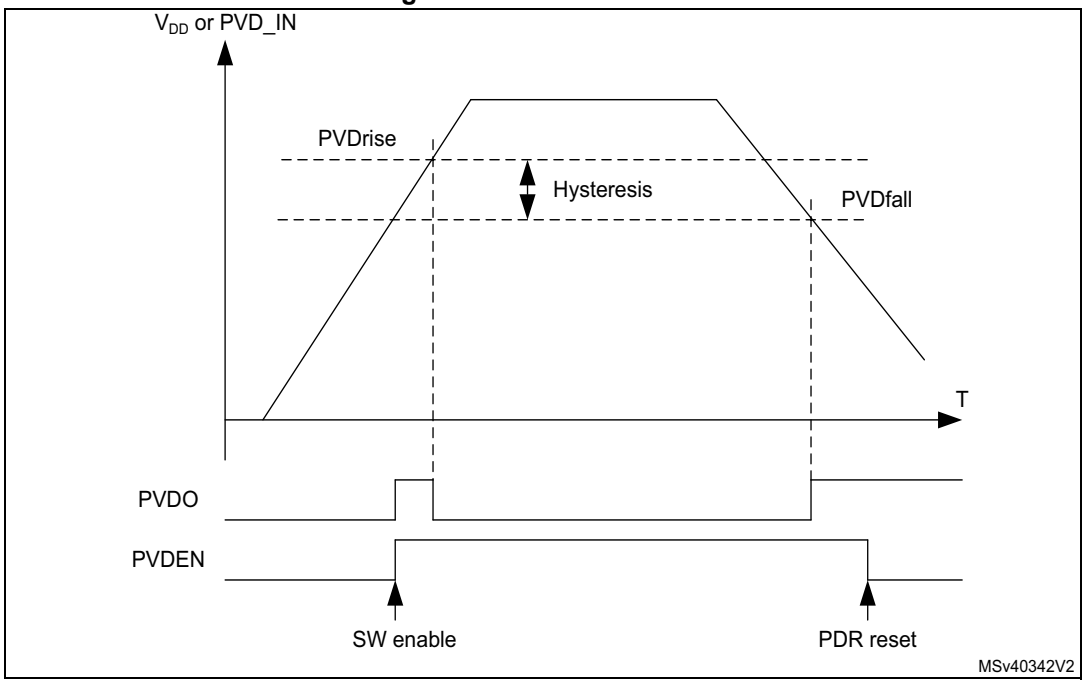
6.5.3 Programmable voltage detector (PVD)

The PVD can be used to monitor the V_{DD} power supply by comparing it to a threshold selected by the PLS[2:0] bits in the *PWR control register 1 (PWR_CR1)*. The PVD can also be used to monitor a voltage level on the PVD_IN pin. In this case the PVD_IN voltage is compared to the internal V_{REFINT} level.

The PVD is enabled by setting the PVDE bit in the *PWR control register 1 (PWR_CR1)*.

A PVDO flag is available in the *PWR status register 1 (PWR_SR1)* to indicate if the V_{DD} or PVD_IN voltage is higher or lower than the PVD threshold. This event is internally connected to the EXTI and can generate an interrupt, assuming it has been enabled through the EXTI registers. The PVDO output interrupt can be generated when V_{DD} or PVD_IN voltage drops below the PVD threshold and/or when V_{DD} or PVD_IN voltage rises above the PVD threshold depending on EXTI rising/falling edge configuration. As an example the service routine could perform emergency shutdown tasks.

Figure 25. PVD thresholds



1. For thresholds and hysteresis values, refer to the datasheets.

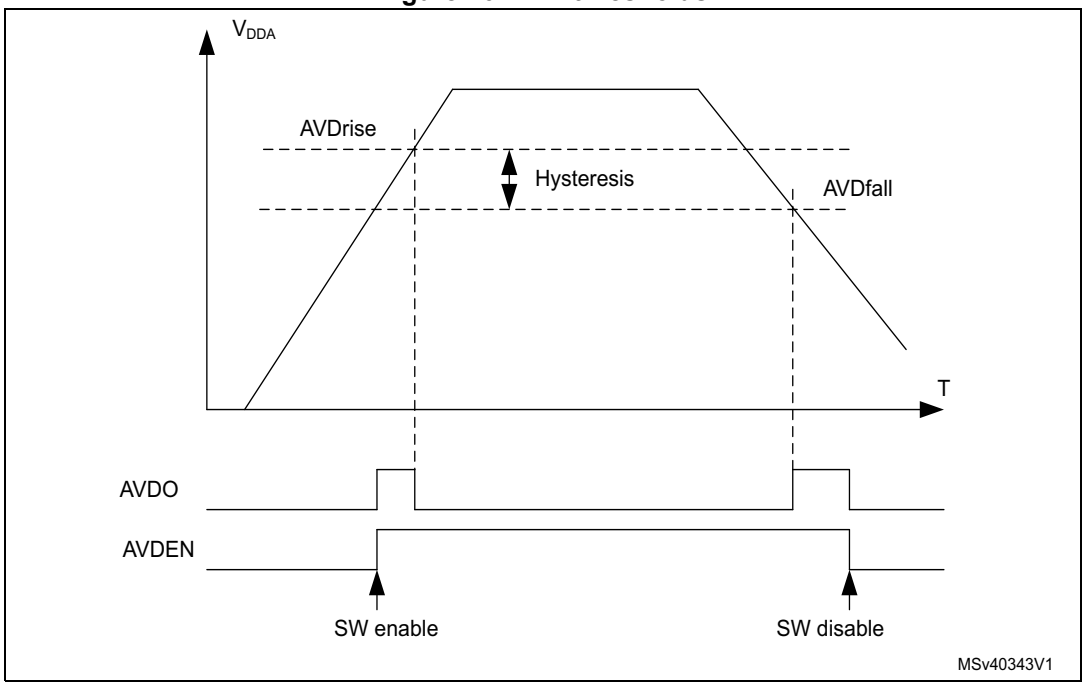
6.5.4 Analog voltage detector (AVD)

The AVD can be used to monitor the V_{DDA} supply by comparing it to a threshold selected by the ALS[1:0] bits in the *PWR control register 1 (PWR_CR1)*.

The AVD is enabled by setting the AVDEN bit in the *PWR control register 1 (PWR_CR1)*.

An AVDO flag is available in the *PWR status register 1 (PWR_SR1)* to indicate whether V_{DDA} is higher or lower than the AVD threshold. This event is internally connected to the EXTI and can generate an interrupt if enabled through the EXTI registers. The AVDO interrupt can be generated when V_{DDA} drops below the AVD threshold and/or when V_{DDA} rises above the AVD threshold depending on the EXTI rising/falling edge configuration. As an example the service routine could indicate when the V_{DDA} supply drops below a minimum level.

Figure 26. AVD thresholds



1. For thresholds and hysteresis values, refer to the datasheets.

6.5.5 Battery voltage thresholds

The battery voltage supply monitors the backup domain V_{SW} level. V_{SW} is monitored by comparing it with two threshold levels: $V_{BAThigh}$ and V_{BATlow}. VBATH and VBATL flags in the *PWR control status register 1 (PWR_CSR1)*, indicate if V_{SW} is higher or lower than the threshold.

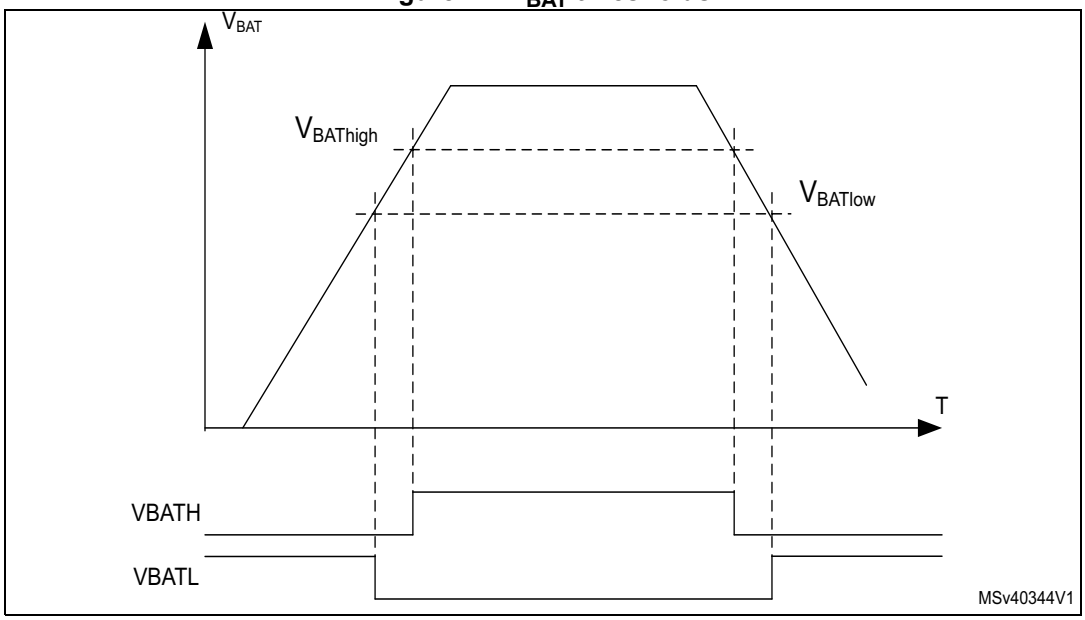
The V_{BAT} supply monitoring can be enabled/disabled via the MONEN bit in the *PWR control status register 1 (PWR_CSR1)*. When it is enabled, the battery voltage thresholds increase power consumption. As an example the V_{SW} levels monitoring could be used to trigger a tamper event for an over or under voltage of the RTC power supply domain (available in V_{BAT} mode).

VBATH and VBATL are connected to RTC tamper signals (see *Section 42: Real-time clock (RTC)*).

Note: Battery voltage monitoring is only available when the backup regulator is enabled (BREN bit set in the PWR control status register 1 (PWR_CSR1)).

When the device does not operate in V_{BAT} mode, the battery voltage monitoring checks the V_{DD} level. When V_{DD} is available, V_{SW} is connected to V_{DD} through the internal power switch (see Section 6.4.3: Backup domain).

Figure 27. V_{BAT} thresholds



1. For thresholds and hysteresis values, refer to the datasheets.

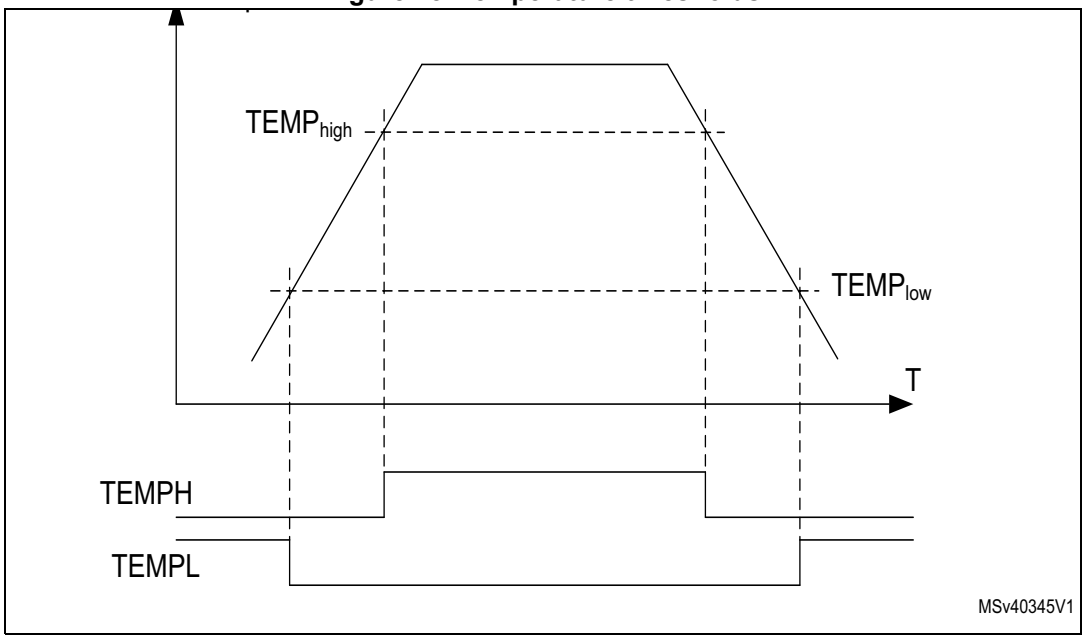
6.5.6 Temperature thresholds

The junction temperature can be monitored by comparing it with two threshold levels, $TEMP_{high}$ and $TEMP_{low}$. TEMPH and TEMPL flags, in the *PWR control status register 1 (PWR_CSR1)*, indicate whether the device temperature is higher or lower than the threshold. The temperature monitoring can be enabled/disabled via the MONEN bit in the *PWR control status register 1 (PWR_CSR1)*. When enabled, the temperature thresholds increase power consumption. As an example the levels could be used to trigger a routine to perform temperature control tasks.

The temperature thresholds are available only when the backup regulator is enabled (BREN bit set in the PWR_CSR1 register).

TEMPH and TEMPL wakeup interrupts are available on the RTC tamper signals (see *Section 42: Real-time clock (RTC)*).

Figure 28. Temperature thresholds



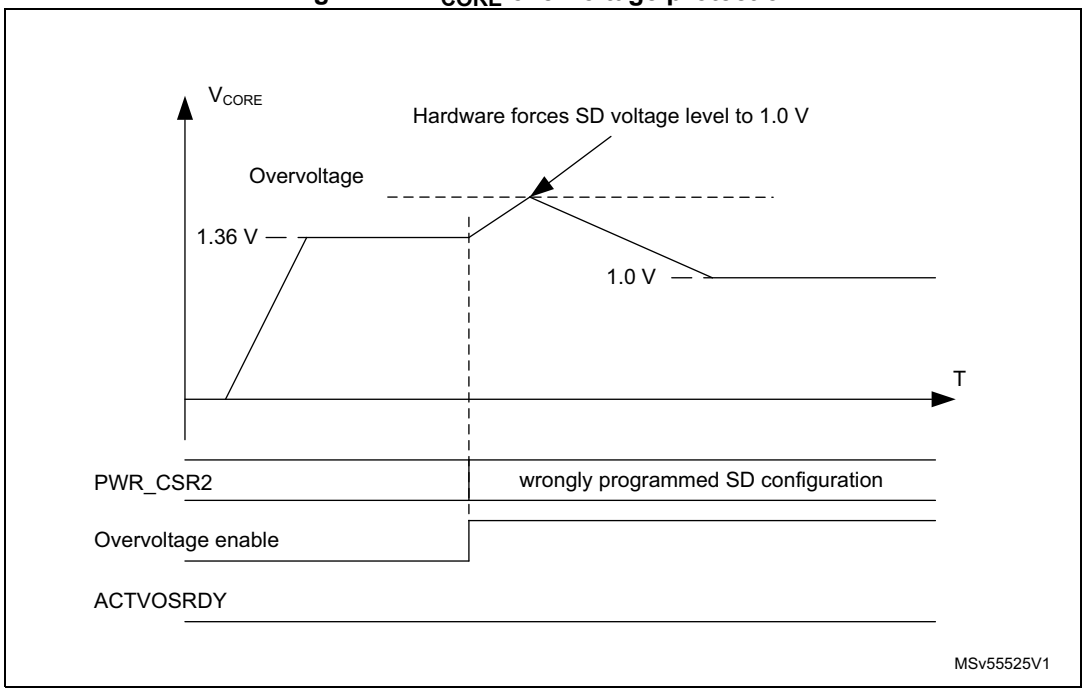
1. For thresholds and hysteresis values, refer to the datasheets.

6.5.7 V_{CORE} maximum voltage level detector

V_{CORE} is protected against too high voltages in the direct SMPS step-down converter supply configuration. V_{CORE} overvoltage protection is enabled at startup by hardware once the SMPS step-down converter configuration has been programmed into *PWR control status register 2 (PWR_CSR2)*:

- V_{CORE} voltage level stays within range:
  - ACTVOSRDY bit in the *PWR status register 1 (PWR_SR1)* indicate valid voltage levels.
  - The system operates normally and V_{CORE} overvoltage protection is disabled.
- V_{CORE} overvoltage (due to a wrongly programmed SMPS step-down converter configuration):
  - The hardware forces the SMPS step-down converter voltage level to 1.0 V.
  - The ACTVOSRDY goes on indicating invalid voltage levels. In this case the software must be corrected and re-loaded to program a correct SMPS step-down converter configuration that matches the application supply connections. The system must then be power cycled.

Figure 29. V_{CORE} overvoltage protection



6.6 Power management

The power management block controls the V_{CORE} supply in accordance with the system operation modes (see *Section 6.6.1*).

The device power domains can operate in one of the following operating modes:

- Run (power ON, clock ON)
- Sleep (power ON, core clock stopped, peripherals keep running)
- Stop (power ON, clock OFF)
- Standby (power OFF, clock OFF).

The V_{CORE} supply level follows the system operating mode (Run, Stop, Standby).

The following voltage scaling features allow controlling the power with respect to the required system performance (see *Section 6.6.2: Voltage scaling*):

- To obtain a given system performance, the corresponding voltage scaling must be set in accordance with the system clock frequency. To do this, configure the VOS bits to the Run mode voltage scaling.
- To obtain the best trade-off between power consumption and exit-from-Stop mode latency, configure the SVOS bit to Stop mode voltage scaling.

6.6.1 Operating modes

Several system operating modes are available to tune the system according to the performance required, that is, when the CPU does not need to execute code and are waiting for an external event. It is up to the user to select the operating mode that gives the best compromise between low-power consumption, short startup time and available wakeup sources.

The operating modes allow controlling the clock distribution to the different system blocks and powering them.

In Run mode, power consumption can be reduced by one of the following means:

- Lowering the system performance by slowing down the system clocks and reducing the V_{CORE} supply level through VOS voltage scaling bit.
- Gating the clocks to the APBx and AHBx peripherals when they are not used, through PERxEN bits.

Table 40. Operating mode summary

System	Entry	Wakeup	System oscillator	System clock	Peripheral clock	CPU clock	Voltage regulator
Run	-	-	ON	ON	ON	ON	ON (VOS)
Sleep	WFI or return from ISR or WFE	Any interrupt or event			ON/OFF(1)	OFF	
Stop	SLEEPDEEP bit + WFI or return from ISR or WFE or Wakeup source cleared(2)	Any EXTI interrupt or event	ON/OFF(3)	OFF			ON (SVOS)
Standby	PDDS bit + SLEEPDEEP bit + WFI or return from ISR or WFE or Wakeup source cleared(2)	WKUP pins rising or falling edge, RTC alarm (Alarm A or Alarm B), RTC Wakeup event, RTC tamper events, RTC time stamp event, external reset in NRST pin, IWDG reset	OFF		OFF		OFF

1. The CPU subsystem peripherals that have a PERxLPEN bit operate accordingly.
2. When the CPU is in Stop the last EXTI Wakeup source should be cleared by SW.
3. When the system oscillator HSI or CSI is used, the state is controlled by HSIKERON and CSIKERON, otherwise the system oscillator is OFF.

6.6.2 Voltage scaling

The V_{CORE} domain is supplied from a single voltage regulator supporting voltage scaling with the following features:

- Run mode voltage scaling
  - VOS2
  - VOS1
  - VOS0
- Stop mode voltage scaling
  - SVOS low
  - SVOS high

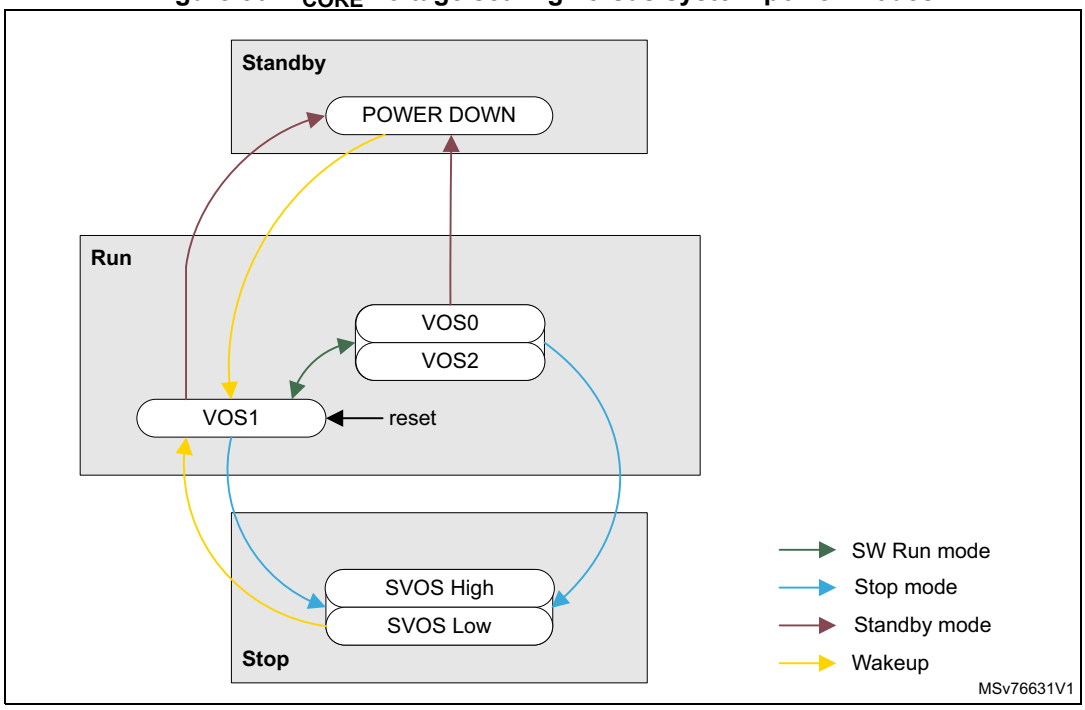
For more details on voltage scaling values, refer to the product datasheet.

After reset, the system starts up at the default Run mode voltage scaling (VOS1). The voltage scaling can then be changed on-the-fly by software by programming the VOS bit field in the *PWR control status register 4 (PWR_CSR4)* according to the required system performance. When exiting from Stop mode or Standby mode, the Run mode voltage scaling is reset to the default VOS1 value.

Before entering Stop mode, the software can preselect the SVOS level in *PWR control register 1 (PWR_CR1)*. The Stop mode voltage scaling for SVOS low and SVOS high also sets the voltage regulator in Low-power (LP) mode to further reduce power consumption.

In Standby mode the VCORE supply is switched off.

Figure 30. V_{CORE} voltage scaling versus system power modes



6.6.3 Power management examples

- *Figure 31* shows V_{CORE} voltage scaling behavior in Run mode.
- *Figure 32* shows V_{CORE} voltage scaling behavior in Stop mode.
- *Figure 33* shows V_{CORE} voltage regulator and voltage scaling behavior in Standby mode.

Example of V_{CORE} voltage scaling behavior in Run mode

Figure 31 illustrates the following system operation sequence example:

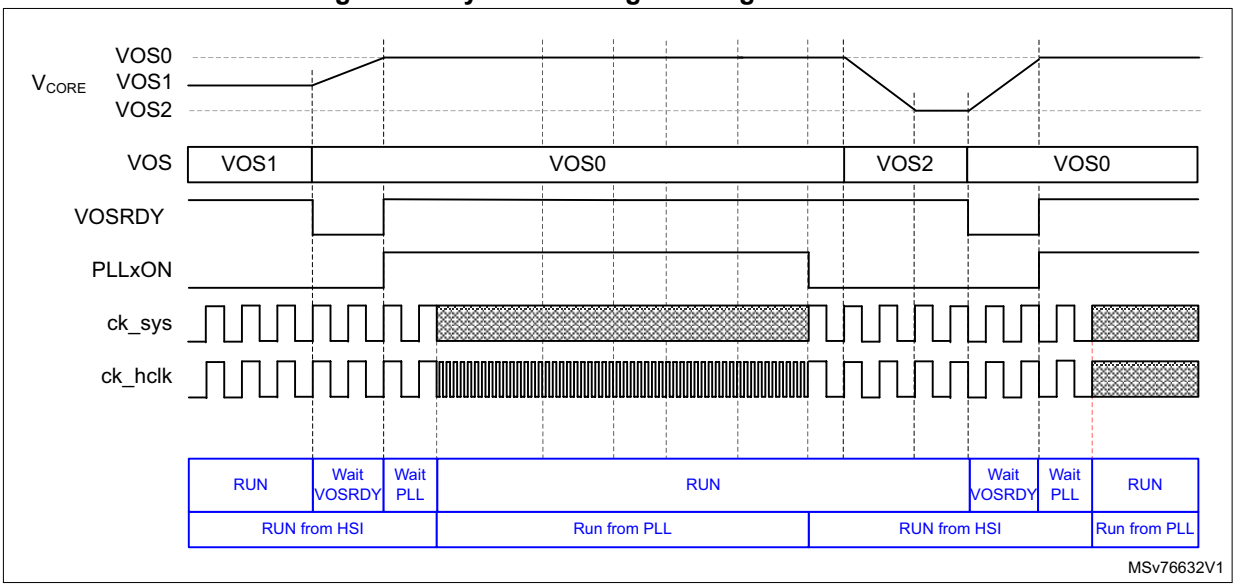
1. After reset, the system starts from HSI with VOS1.
2. The system performance is then increased to a high-speed clock from the PLL with voltage scaling VOS0. To do this:
   a) Program the voltage scaling to VOS0.
   b) Once the V_{CORE} supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.
3. The next step is to reduce the system performance to HSI clock with voltage scaling VOS2. To do this:
   a) Switch the clock to HSI.
   b) Disable the PLL.

c) Decrease the voltage scaling to VOS2.

4. The system performance can then be increased to high-speed clock from the PLL. To do this:
   a) Program the voltage scaling to VOS0.
   b) Once the V_{CORE} supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.

When the system performance (clock frequency) is changed, VOS must be set accordingly. Otherwise the system might be unreliable.

Figure 31. Dynamic voltage scaling in Run mode



1. The status of the register bits at each step is shown in blue.

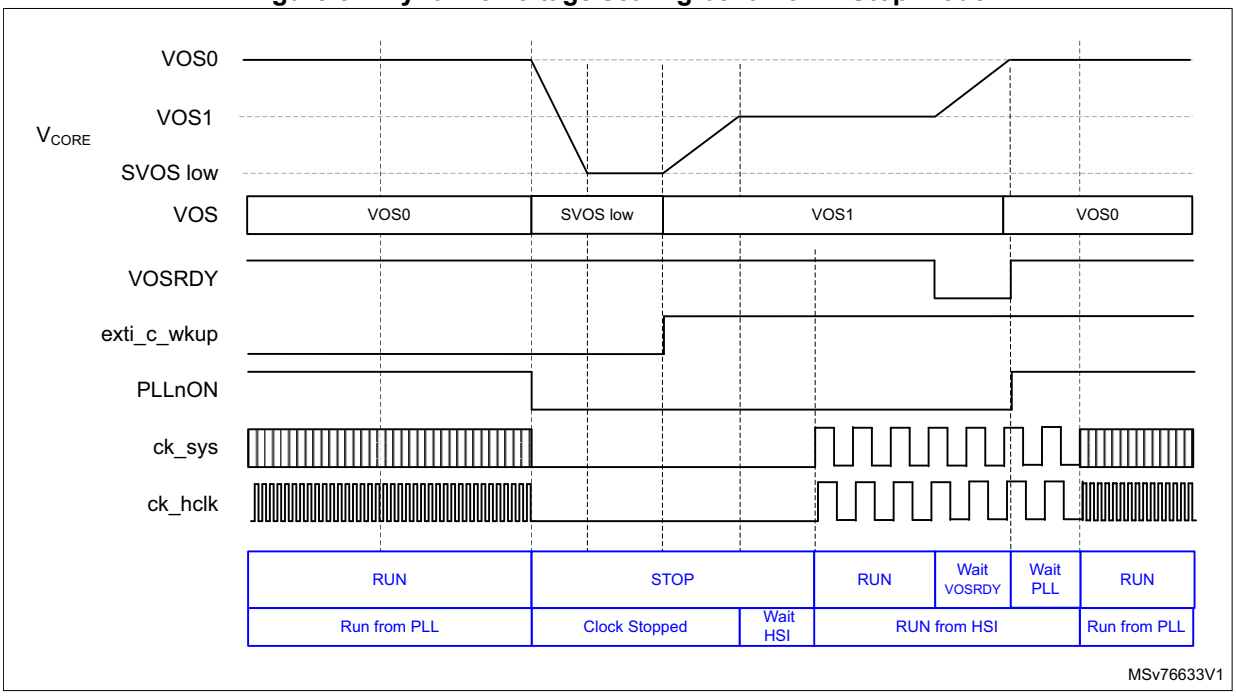
Example of V_{CORE} voltage scaling behavior in Stop mode

Figure 32 illustrates the following system operation sequence example:

1. The system is running from the PLL in high-performance mode (VOS0 voltage scaling).
2. First the CPU subsystem deallocates all the peripheral, in a second step, the CPU subsystem enters Stop mode and the system enters Stop mode. The system clock is stopped and the hardware lowers the voltage scaling to the software preselected SVOS low.

3. The CPU subsystem is then woken up. The system exits from Stop mode and the CPU subsystem exits from Stop mode. The hardware always sets the voltage scaling to VOS1 after exiting from Stop mode and waits for the requested supply level to be reached before enabling the HSI clock. Once the HSI clock is stable, the system clock is enabled and switch in run mode under HSI clock.
4. The system performance could then be increased. To do this:
   a) The software first sets the voltage scaling to VOS0.
   b) Once the V_{CORE} supply has reached the required level indicated by VOSRDY, the clock frequency can be increased by enabling the PLL.
   c) Once the PLL is locked, the system clock can be switched.

Figure 32. Dynamic voltage scaling behavior in Stop mode



1. The status of the register bits at each step is shown in blue.

Example of V_{CORE} voltage regulator and voltage scaling behavior in Standby mode

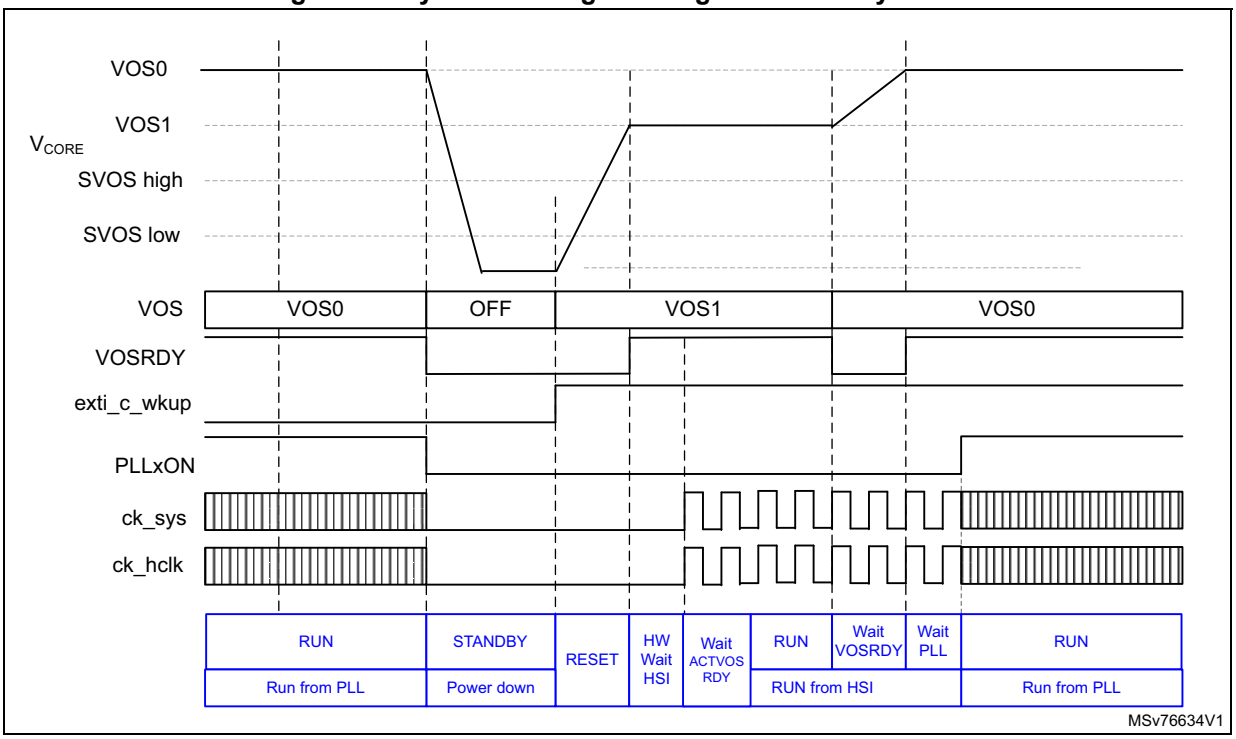
Figure 33 illustrates the following system operation sequence example:

1. The system is running from the PLL in high-performance mode (VOS0 voltage scaling).
2. The CPU subsystem deallocates all the peripherals and the power is switched off. The system enters in standby mode.
3. The system is then woken up by a wakeup source. The system exits from Standby mode. The hardware sets the voltage scaling to the default VOS1 and waits for the requested supply level to be reached before enabling the default HSI oscillator. Once

the HSI clock is stable, the system clock is enabled. The software must then check the ACTVOSRDY is valid before changing the system performance.

4. In a next step, increase the system performance. To do this:
   a) The software first increases the voltage scaling to VOS0.
   b) Before enabling the PLL, it waits for the requested supply level to be reached by monitoring VOSRDY bit.
   c) Once the PLL is locked, the system clock can be switched.

Figure 33. Dynamic voltage scaling from Standby mode



1. The status of the register bits at each step is shown in blue.

6.7 Low-power modes

Several low-power modes are available to save power when the CPU does not need to execute code (i.e. when waiting for an external event). It is up to the user application to select the mode that gives the best compromise between low-power consumption, short startup time and available wakeup sources:

- Slowing down system clocks (see *Section 7.5.6: System clock (sys_ck)*)
- Controlling individual peripheral clocks (see *Section 7.5.12: Peripheral clock gating control*)
- Low-power modes
  - Sleep (CPU clock stopped and still in Run mode)
  - Stop (System clock stopped)
  - Standby (System powered down)

6.7.1 Slowing down system clocks

In Run mode, the speed of the system clock sys_ck can be reduced. For more details refer to *Section 7.5.6: System clock (sys_ck)*.

6.7.2 Controlling peripheral clocks

In Run mode, the HCLKx and PCLKx for individual peripherals can be stopped by configuring at any time PERxEN bit in RCC_xxxxENR to reduce power consumption.

To reduce power consumption in CSleep mode, the individual peripheral clocks can be disabled by configuring PERxLPEN bit in RCC_xxxxLPENR. For the peripherals still receiving a clock in Sleep mode, their clock can be slowed down before entering Sleep mode.

Example: for APB2 peripherals: the RCC register is named: RCC_APB2ENR, this register contains SPI4EN and so on.

6.7.3 Entering low-power modes

CPU subsystem Sleep and Stop low-power modes are entered by the MCU when executing the WFI (Wait For Interrupt) or WFE (wait for event) instructions, or when the SLEEPONEXIT bit in the Cortex®-M system control register is set on return from ISR.

The system can enter Stop or Standby low-power mode when all EXTI wakeup sources are cleared.

6.7.4 Exiting from low-power modes

The CPU subsystem exits from Sleep mode through any interrupt or event depending on how the low-power mode was entered:

- If the WFI instruction or return from ISR was used to enter to low-power mode, any peripheral interrupt acknowledged by the NVIC can wake up the system.
- If the WFE instruction is used to enter to low-power mode, the CPU exits from low-power mode as soon as an event occurs. The wakeup event can be generated either by:
  - An NVIC IRQ interrupt.

    When SEVONPEND = 0 in the Cortex®-M7 system control register, the interrupt must be enabled in the peripheral control register and in the NVIC.

    When the MCU resumes from WFE, the peripheral interrupt pending bit and the NVIC peripheral IRQ channel pending bit in the NVIC interrupt clear pending register have to be cleared. Only NVIC interrupts with sufficient priority wake up and interrupts the MCU.

    When SEVONPEND = 1 in the Cortex®-M7 system control register, the interrupt must be enabled in the peripheral control register and optionally in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and, when

enabled, the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.

All NVIC interrupts wakeup the MCU, even the disabled ones.

Only enabled NVIC interrupts with sufficient priority wakeup and interrupt the MCU.

– An event

An EXTI line must be configured in event mode. When the CPU resumes from WFE, it is not necessary to clear the EXTI peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bits corresponding to the event line is not set. It might be necessary to clear the interrupt flag in the peripheral.

The CPU subsystem exits from Stop modes by enabling an EXTI interrupt or event depending on how the low-power mode was entered (see above).

The CPU subsystem exits from Standby mode by enabling an EXTI interrupt or event. Program execution restarts from CPU local reset (such as a reset vector fetched from the system configuration boot and security block (SBS).

The CPU subsystem exits from Standby mode by enabling an external reset (NRST pin), an IWDG reset, a rising edge on one of the enabled WKUPx pins or an RTC event. Program execution restarts in the same way as after a system reset (such as boot pin sampling, option bytes loading or reset vector fetched).

6.7.5 Sleep mode

The Sleep mode applies only to the CPU subsystem. In Sleep mode, the CPU clock is stopped. The CPU subsystem peripheral clocks operate according to the values of PERxLPEN bits in RCC_xxxxENR.

Entering Sleep mode

The Sleep mode is entered according to *Section 6.7.3: Entering low-power modes*, when the SLEEPDEEP bit in the Cortex®-M System Control register is cleared.

Refer to *Table 41* for details on how to enter to Sleep mode.

Exiting from Sleep mode

The Sleep mode is exited according to *Section 6.7.4: Exiting from low-power modes*.

Refer to *Table 41* for more details on how to exit from Sleep mode.

Table 41. Sleep mode

Sleep mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0 (refer to the Cortex®-M System Control register.) – CPU NVIC interrupts and events cleared.
	On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 (refer to the Cortex®-M System Control register.) – CPU NVIC interrupts and events cleared.
Mode exit	If WFI or return from ISR was used for entry: – Any Interrupt enabled in NVIC: Refer to *Table 110: NVIC* If WFE was used for entry and SEVONPEND = 0: – Any event: Refer to *Section 16.5.3: EXTI CPU wake-up procedure* If WFE was used for entry and SEVONPEND = 1: – Any Interrupt even when disabled in NVIC: refer to *Table 110: NVIC* or any event: refer to *Section 16.5.3: EXTI CPU wake-up procedure*
Wakeup latency	None

6.7.6 Stop mode

The Stop mode applies to the MCU. In Stop mode, the CPU clock is stopped. All CPU subsystem peripheral clocks are stopped too.

In Stop mode, CPU subsystem peripherals having a kernel clock request can still request their kernel clock.

The flash memory can enter low-power Stop mode when it is enabled through FLPS in PWR_CR1 register. This allows a trade-off between domain Stop restart time and low-power consumption.

The HSI or CSI can remain enabled in system Stop mode (HSIKERON and CSIKERON set in RCC_CR register). After exiting Stop mode, the clock is quickly available as kernel clock

for peripherals. Other system oscillator sources are stopped in Stop mode and require a starting time after exiting Stop mode.

In Stop mode and SVOS high, peripherals using the LSI or LSE clock and peripherals having a kernel clock request are still able to operate.

In system Stop mode, the following features can be selected to remain active by programming individual control bits:

- Independent watchdog (IWDG)

  The IWDG is started by writing to its key register or by hardware option. Once started it cannot be stopped except by a reset (see *Section 40.4* in *Section 40: Independent watchdog (IWDG)*.
- Real-time clock (RTC)

  This is configured via the RTCEN bit in the *RCC Backup domain control register (RCC_BDCR)*.
- Internal RC oscillator (LSI RC)

  This is configured via the LSION bit in the *RCC clock control and status register (RCC_CSR)*.
- External 32.768 kHz oscillator (LSE OSC)

  This is configured via the LSEON bit in the *RCC Backup domain control register (RCC_BDCR)*.
- Peripherals capable of running on the LSI or LSE clock.
- Peripherals having a kernel clock request.
- Internal RC oscillators (HSI and CSI)

  This is configured via the HSIKERON and CSIKERON bits in the *RCC source control register (RCC_CR)*.
- The ADC can also consume power during Stop mode, unless it is disabled before entering this mode. To disable them, the ADON bit in the ADC_CR2 register must be written to 0.

The selected SVOS low levels add an additional startup delay when exiting from system Stop mode (see *Table 42*).

Table 42. Stop mode operation

SVOS	Stop mode Voltage regulator operation	Wake-up Latency
SVOS high	**LP**	Voltage regulator wakeup time from SVOS high LP mode to VOS low
SVOS low	**LP**	Voltage regulator wakeup time from SVOS low LP mode to VOS low

Entering Stop mode

The Stop mode is entered according to *Section 6.7.3: Entering low-power modes*, when the SLEEPDEEP bit in the Cortex®-M system control register is set.

If flash memory programming is ongoing, the Stop mode entry is delayed until the memory access is finished.

If an access to the domain bus matrix is ongoing, the Stop mode entry is delayed until the domain bus matrix access is complete.

See *Table 43* below for details on how to enter to Stop mode.

Table 43. Stop mode

Stop mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP = 1 (refer to the Cortex®-M System control register.) – CPU NVIC interrupts and events cleared. – All CPU EXTI wakeup sources are cleared.
	On return from ISR while: – SLEEPDEEP = 1 and – SLEEPONEXIT = 1 (refer to the Cortex®-M system control register.) – CPU NVIC interrupts and events cleared. – All CPU EXTI wakeup sources are cleared.
Mode exit	If WFI or return from ISR was used for entry: – EXTI interrupt enabled in NVIC: refer to *Table 110: NVIC*, for peripheral which are not stopped or powered down. If WFE was used for entry and SEVONPEND = 0: – EXTI event: refer to *Section 16.5.3: EXTI CPU wake-up procedure*, for peripheral which are not stopped or powered down. If WFE was used for entry and SEVONPEND = 1: – EXTI interrupt even when disabled in NVIC: refer to *Table 110: NVIC* or EXTI event: refer to *Section 16.5.3: EXTI CPU wake-up procedure*, for peripheral which are not stopped or powered down.
Wakeup latency	EXTI and RCC wakeup synchronization (see *Section 7.4.11: Power-on and wakeup sequences*)

To allow peripherals having a kernel clock request to operate in Stop mode, the system must use SVOS high.

Note: Use a DSB instruction to ensure that outstanding memory transactions complete before entering stop mode.

Exiting from Stop mode

The Stop mode is exited according to *Section 6.7.4: Exiting from low-power modes*.

Refer to *Table 43: Stop mode* for more details on how to exit from Stop mode.

When exiting from Stop mode, the system clock, bus matrix clocks and voltage scaling are reset.

STOPF status flag in the *PWR CPU control status register 3 (PWR_CSR3)* indicates that the system has exited from Stop mode.

I/O states in Stop mode

I/O pin configuration remains unchanged in Stop mode.

6.7.7 Standby mode

The Standby mode allows achieving the lowest power consumption. Like Stop mode, it is based on CPU subsystem Stop mode. However the V_{CORE} supply regulator is powered off.

When the system enters in Standby mode, the voltage regulator is disabled. The complete V_{CORE} domain is consequently powered off. The PLLs, HSI oscillator, CSI oscillator, HSI48 and the HSE oscillator are also switched off. SRAM and register contents are lost except for backup domain registers (RTC registers, RTC backup register and backup RAM), and Standby circuitry (see *Section 6.4.3: Backup domain*).

In system Standby mode, the following features can be selected by programming individual control bits:

- Independent watchdog (IWDG)

  The IWDG is started by programming its key register or by hardware option. Once started, it cannot be stopped except by a reset (see *Section 40.4* in *Section 40: Independent watchdog (IWDG)*.
- Real-time clock (RTC)

  This is configured via the RTCEN bit in the backup domain control register (RCC_BDCR).
- Internal RC oscillator (LSI RC)

  This is configured by the LSION bit in the RCC clock control and status register (RCC_CSR).
- External 32.768 kHz oscillator (LSE OSC)

  This is configured by the LSEON bit in the backup domain control register (RCC_BDCR).

Entering Standby mode

The Standby mode is entered according to *Section 6.7.3: Entering low-power modes*, when the PDDS bits in the *PWR CPU control status register 3 (PWR_CSR3)* are set to one.

If flash memory programming is ongoing, the Standby mode entry is delayed until the memory access is finished.

Refer to *Table 45* for more details on how to enter to Standby mode.

Exiting from Standby mode

The Standby mode is exited according to *Section 6.7.4: Exiting from low-power modes*.

The system exits from Standby mode when an external reset (NRST pin), an IWDG reset, a WKUP pin event, an RTC alarm, a tamper event, or a time stamp event is detected. All registers are reset after waking up from Standby except for power control and status registers (*PWR control status register 2 (PWR_CSR2)*, *PWR CPU control status register 3 (PWR_CSR3)*), SBF bit in the *PWR CPU control status register 3 (PWR_CSR3)*, *PWR wakeup flag register (PWR_WKUPFR)*, and *PWR wakeup enable and polarity register (PWR_WKUPEPR)*.

After waking up from Standby mode, the program execution restarts in the same way as after a system reset (boot option sampling, boot vector reset fetched, etc.). The SBF status flags in the *PWR CPU control status register 3 (PWR_CSR3)* registers indicate from which mode the system has exited (refer to *Table 45*).

Table 44. Standby and Stop flags

SBF	STOPF	Description
0	1	System has been in Stop
1	0	System has been in Standby

Table 45. Standby mode

Standby mode	Description
Mode entry	– The CPU subsystem is in Stop mode, and there is no active EXTI wakeup source. – PDDS bit for select Standby. – All WKUPF bits in the PWR wakeup flag register (PWR_WKUPFR) are cleared.
Mode exit	– WKUP pins rising or falling edge, RTC alarm (Alarm A and Alarm B), RTC wakeup, tamper event, time stamp event, external reset in NRST pin, IWDG reset.
Wakeup latency	System reset phase (see *Section 7.4.2: The system and application resets (sys_rst and nreset)*)

I/O states in Standby mode

In Standby mode, all I/O pins are high impedance without pull, except for:

- Reset pad (still available)
- RTC_AF1 pin if configured for tamper, time stamp, RTC Alarm out, or RTC clock calibration out
- WKUP pins (if enabled). The WKUP pin pull configuration can be defined through WKUPPUPD register bits in the *PWR wakeup enable and polarity register (PWR_WKUPEPR)*
- I3C pin pull-up as configured in *PWR I3C pull-up control register (PWR_I3CPUCR)*, if enabled by the APC bit in *PWR apply pull register (PWR_APCR)*.
- XSPI control pin pull-up/down as configured in *PWR XSPI pull-up control register (PWR_XSPIPUCR)*, and *PWR XSPI pull-down control register (PWR_XSPIPDCR)*, if enabled by the APC bit in *PWR apply pull register (PWR_APCR)*

6.7.8 Monitoring low-power modes

The devices feature state monitoring pins to monitor the state transition to low-power mode (refer to *Table 46* for the list of pins and their description). The GPIO pin corresponding to each monitoring signal has to be programmed in alternate function mode.

This feature is not available in Standby mode since these I/O pins are switched to high impedance.

Table 46. Low-power modes monitoring pin overview

Power state monitoring pins	Description
PWR_CSLEEP (CSLEEP)	Sleeping CPU state
PWR_CSTOP (CDSLEEP)	Deepsleep CPU state

The values of the monitoring pins reflect the state of the CPU and domains. Refer to *Table 47* for the GPIO state depending on CPU and domain state.

Table 47. GPIO state according to CPU and domain state

CPU		CPU power state	Power mode
CSLEEP	CDSLEEP		
0	0	CPU in Run mode	Run mode
1	0	CPU in Sleep mode	
0	1	CPU in Run mode	
1	1	CPU in Deepsleep mode	Stop mode
-	-	-(1)	Standby mode

1. The CPU is powered off.

6.8 PWR registers

The PWR registers can be accessed in word, half-word and byte format, unless otherwise specified.

6.8.1 PWR control register 1 (PWR_CR1)

Address offset: 0x000

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALS[1:0]		AVDEN	AVDREADY	BOOSTE	Res.	FLPS	DBP	PLS[2:0]			PVDE	Res.	Res.	Res.	SVOS
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw				rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **ALS[1:0]**: Analog voltage detector level selection

These bits select the voltage threshold detected by the AVD.

00: AVD level 1

01: AVD level 2

10: AVD level 3

11: AVD level 4

Note: Refer to Section "Electrical characteristics" of the product datasheet for more details.

Bit 13 **AVDEN**: Peripheral voltage monitor on V_{DDA} enable

0: Peripheral voltage monitor on V_{DDA} disabled

1: Peripheral voltage monitor on V_{DDA} enabled

Bit 12 **AVDREADY**: Analog voltage ready

This bit is only used when the analog switch boost needs to be enabled (see BOOSTE bit).

It must be set by software when the expected $_{VDD}$A analog supply level is available.

The correct analog supply level is indicated by the AVDO bit (PWR_CSR1 register) after setting the AVDEN bit and selecting the supply level to be monitored (ALS bits).

0: peripheral analog voltage V_{DDA} not ready (default)

1: peripheral analog voltage V_{DDA} ready

Bit 11 **BOOSTE**: Analog switch VBoost control

This bit enables the booster to guarantee the analog switch AC performance when the V_{DD} supply voltage is below 2.7 V (reduction of the total harmonic distortion to have the same switch performance over the full supply voltage range).

The V_{DD} supply voltage can be monitored through the PVD and the PLS bits.

0: booster disabled (default)

1: booster enabled if analog voltage ready (AVD_READY = 1)

Bit 10 Reserved, must be kept at reset value.

Bit 9 **FLPS**: Flash low-power mode in Stop mode

This bit allows to obtain the best trade-off between low-power consumption and restart time when exiting from Stop mode.

When it is set, the flash memory enters low-power mode when device is in Stop mode.

0: Flash memory remains in normal mode when device enters Stop (quick restart time).

1: Flash memory enters low-power mode when device enters Stop mode (low-power consumption).

Bit 8 **DBP**: Disable backup domain write protection

In reset state, the RCC_BDCR register, the RTC registers (including the backup registers), the backup SRAM, and the BREN and MONEN bits in the PWR_CSR1 register, are protected against parasitic write access.

This bit must be set to enable access to these registers.

0: Read and write access to RTC and RTC backup registers disabled and backup SRAM write access disabled

1: Read and write access to RTC, RTC backup registers and backup SRAM enabled

Bits 7:5 **PLS[2:0]**: Programmable voltage detector level selection

These bits selects the condition below which the PVD output (PVDO bit in *PWR status register 1 (PWR_SR1)*) goes high:

000: V_{DD} < PVD level 1

001: V_{DD} < PVD level 2

010: V_{DD} < PVD level 3

011: V_{DD} < PVD level 4

100: V_{DD} < PVD level 5

101: V_{DD} < PVD level 6

110: V_{DD} < PVD level 7

111: PVD_IN < VREFINT

Note: Refer to Section "Electrical characteristics" of the product datasheet for more details.

Bit 4 **PVDE**: Programmable voltage detector enable

0: Programmable voltage detector disabled.

1: Programmable voltage detector enabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **SVOS**: System Stop mode voltage scaling selection

0: SVOS low

1: SVOS high (default)

6.8.2 PWR status register 1 (PWR_SR1)

Address offset: 0x004

Reset value: 0x0000 0002

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

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **AVDO**: Analog voltage detector output on V_{DDA}

This bit is set and cleared by hardware. It is valid only if AVD on V_{DDA} is enabled by the AVDEN bit.

0: V_{DDA} is equal or higher than the AVD threshold selected with the ALS[1:0] bits.

1: V_{DDA} is lower than the AVD threshold selected with the ALS[1:0] bits

Note: Since the AVD is disabled in Standby mode, this bit is equal to 0 after Standby or reset until the AVDEN bit is set

Bits 12:5 Reserved, must be kept at reset value.

Bit 4 **PVDO**: Programmable voltage detect output

This bit is set and cleared by hardware. It is valid only if the PVD has been enabled by the PVDE bit.

0: V_{DD} or PVD_IN voltage is equal or higher than the PVD threshold selected through the PLS[2:0] bits.

1: V_{DD} or PVD_IN voltage is lower than the PVD threshold selected through the PLS[2:0] bits.

Note: Since the PVD is disabled in Standby mode, this bit is equal to 0 after Standby or reset until the PVDE bit is set.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **ACTVOSRDY**: Voltage levels ready bit for currently used ACTVOS

This bit is set to 1 by hardware when the voltage regulator and the SMPS step-down converter are both disabled and Bypass mode is selected in PWR control register 2 (PWR_CSR2).

0: Voltage level invalid, above or below current ACTVOS selected levels.

1: Voltage level valid, at current ACTVOS selected levels.

Bits 1:0 **ACTVOS[1:0]**: VOS currently applied for V_{CORE} voltage scaling selection

These bit reflect the last VOS value applied to the PMU.

00: Reserved

01: VOS2

10: VOS1

11: VOS0

6.8.3 PWR control status register 1 (PWR_CSR1)

Address offset: 0x008

Reset value: 0x0000 0000

This register is not reset by wakeup from Standby mode, RESET signal and V_{DD} POR. It is only reset by V_{SW} POR and VSWRST reset.

This register must not be accessed when the VSWRST bit in the RCC_BDCR register resets the V_{SW} domain.

After reset, the PWR_CSR1 register is write-protected. Prior to modifying its content, the DBP bit in the PWR_CR1 register must be set to disable the write protection.

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

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TEMPH**: Temperature level monitoring versus high threshold
0: Temperature below high threshold level.
1: Temperature equal or above high threshold level.

Bit 22 **TEMPL**: Temperature level monitoring versus low threshold
0: Temperature above low threshold level.
1: Temperature equal or below low threshold level.

Bit 21 **VBATH**: V_{BAT} level monitoring versus high threshold
0: V_{BAT} level below high threshold level.
1: V_{BAT} level equal or above high threshold level.

Bit 20 **VBATL**: V_{BAT} level monitoring versus low threshold
0: V_{BAT} level above low threshold level.
1: V_{BAT} level equal or below low threshold level.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **BRRDY**: Backup regulator ready
This bit is set by hardware to indicate that the backup regulator is ready.
0: Backup regulator not ready.
1: Backup regulator ready.

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **MONEN:** V_{BAT} and temperature monitoring enable
When set, the V_{BAT} supply and temperature monitoring is enabled.
0: V_{BAT} and temperature monitoring disabled.
1: V_{BAT} and temperature monitoring enabled.
Note: V_{BAT} and temperature monitoring are only available when the backup regulator is enabled (BREN bit set to 1).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **BREN:** Backup regulator enable
When set, the backup regulator (used to maintain the backup RAM content in Standby and V_{BAT} modes) is enabled.
If BREN is reset, the backup regulator is switched off. The backup RAM can still be used in Run and Stop modes. However, its content is lost in Standby and V_{BAT} modes.
If BREN is set, the application must wait till the backup regulator ready flag (BRRDY) is set to indicate that the data written into the SRAM is maintained in Standby and V_{BAT} modes.
0: Backup regulator disabled.
1: Backup regulator enabled.

6.8.4 PWR control status register 2 (PWR_CSR2)

Address offset: 0x00C

Reset value: 0x0000 0006

This register is reset only by a power-on reset (POR). It is not reset by a wakeup from Standby mode or RESET signal.

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

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SDRDY**: Integrated SMPS step-down converter supply ready
This bit is set by hardware to indicate that the supply from the SMPS step-down converter is ready.
0: SMPS supply not ready.
1: SMPS supply ready.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **VBRS**: V_{BAT} charging resistor selection
0: Charge V_{BAT} through a 5 kΩ resistor.
1: Charge V_{BAT} through a 1.5 kΩ resistor.

Bit 8 **VBE**: V_{BAT} charging enable
0: V_{BAT} battery charging disabled.
1: V_{BAT} battery charging enabled.

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

Bit 2 **SMPSEN**: Integrated SMPS step-down converter enable (read only)
This bit is valid only when BYPASS has been programmed once after power-up.
0: SMPS step-down converter disabled
1: SMPS step-down converter enabled

Bit 1 **LDOEN**: Integrated low drop-out regulator enable (read only)
This bit is valid only when BYPASS has been programmed once after power-up.
0: Low drop-out regulator disabled
1: Low drop-out regulator enabled

Bit 0 **BYPASS**: Power management unit bypass
This bit must be programmed after power-up to select the V_{CORE} supply configuration (internal or external). After programming it can not be changed without a power down reset.
0: Internal supply (LDO or SMPS)
1: External supply.

6.8.5 PWR CPU control status register 3 (PWR_CSR3)

This register allows controlling the CPU power.

Address offset: 0x010

Reset value: 0x0000 0000

After Standby Reset value: 0x0000 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	SBF	STOPF	Res.	Res.	Res.	Res.	Res.	Res.	CSSF	PDDS
						r	r							rc_w1	rw

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

Bit 9 **SBF**: System Standby flag
This bit is set by hardware and cleared only by a POR (power-on reset) or by setting the CPU CSSF bit.
0: System has not been in Standby mode
1: System has been in Standby mode

Bit 8 **STOPF**: Stop flag
This bit is set by hardware and cleared only by any reset or by setting the CPU CSSF bit.
0: System has not been in Stop mode
1: System has been in Stop mode

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

Bit 1 **CSSF**: Clear Standby and Stop flags (always read as 0)
This bit is cleared to 0 by hardware.
0: No effect.
1: flags (STOPF, SBF) are cleared.

Bit 0 **PDDS**: Power-down Deepsleep.
This bit allows CPU to define the Deepsleep mode
0: Stop mode when device enters Deepsleep.
1: Standby mode when device enters Deepsleep.

6.8.6 PWR control status register 4 (PWR_CSR4)

Address offset: 0x014

Reset value: 0x0000 0006

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

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

Bit 2 **VOSRDY**: VOS ready bit for V_{CORE} voltage scaling output selection.
When an internal regulator is used, this bit indicates that all the features allowed by the selected VOS can be used.
Note: When Bypass mode is selected in the PWR control register (PWR_CSR2), VOSRDY bit is set to 1 by hardware whatever the V_{CORE} level.
0: Not ready, voltage level below VOS selected level.
1: Ready, voltage level at or above VOS selected level.

Bits 1:0 **VOS[1:0]**: Voltage scaling selection according to performance
When an internal regulator is used, these bits control the V_{CORE} voltage level to obtain the best trade-off between power consumption and performance:
- When increasing the performance, the voltage scaling must be changed before increasing the system frequency.
- When decreasing performance, the system frequency must first be decreased before changing the voltage scaling.

00: Reserved
01: VOS2 (lowest power)
10: VOS1 (default)
11: VOS0 (highest frequency)
Note: Refer to section "Electrical characteristics" of the product datasheet for more details.
Note: When Bypass mode is selected in the PWR control register (PWR_CSR2), these bits have no effect

6.8.7 PWR wakeup clear register (PWR_WKUPCR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPC8	WKUPC7	WKUPC6	WKUPC5	WKUPC4	WKUPC3	WKUPC2	WKUPC1
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bit 7 **WKUPC8**: Clear wakeup pin flag for WKUP8
These bits are always read as 0.
0: No effect
1: Writing 1 clears the WKUPF8 flag (bit is cleared to 0 by hardware)

Bit 6 **WKUPC7**: Clear wakeup pin flag for WKUP7
These bits are always read as 0.
0: No effect
1: Writing 1 clears the WKUPF7 flag (bit is cleared to 0 by hardware)

Bit 5 **WKUPC6**: Clear wakeup pin flag for WKUP6
These bits are always read as 0.
0: No effect
1: Writing 1 clears the WKUPF6 flag (bit is cleared to 0 by hardware)

Bit 4 **WKUPC5**: Clear wakeup pin flag for WKUP5
These bits are always read as 0.
0: No effect
1: Writing 1 clears the WKUPF5 flag (bit is cleared to 0 by hardware)

Bit 3 **WKUPC4**: Clear wakeup pin flag for WKUP4
These bits are always read as 0.
0: No effect
1: Writing 1 clears the WKUPF4 flag (bit is cleared to 0 by hardware)

Bit 2 **WKUPC3**: Clear wakeup pin flag for WKUP3
These bits are always read as 0.
0: No effect
1: Writing 1 clears the WKUPF3 flag (bit is cleared to 0 by hardware)

Bit 1 **WKUPC2**: Clear wakeup pin flag for WKUP2
These bits are always read as 0.
0: No effect
1: Writing 1 clears the WKUPF2 flag (bit is cleared to 0 by hardware)

Bit 0 **WKUPC1**: Clear wakeup pin flag for WKUP1
These bits are always read as 0.
0: No effect
1: Writing 1 clears the WKUPF1 flag (bit is cleared to 0 by hardware)

6.8.8 PWR wakeup flag register (PWR_WKUPFR)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF8	WKUPF7	WKUPF6	WKUPF5	WKUPF4	WKUPF3	WKUPF2	WKUPF1
								r	r	r	r	r	r	r	r

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

Bit 7 **WKUPF8**: Wakeup pin WKUP8 flag

This bit is set by hardware and cleared only by a reset pin or by setting the WKUPC8 bit in the PWR wakeup clear register (PWR_WKUPCR).

0: No wakeup event occurred

1: A wakeup event was received from WKUP8 pin

Bit 6 **WKUPF7**: Wakeup pin WKUP7 flag

This bit is set by hardware and cleared only by a reset pin or by setting the WKUPC7 bit in the PWR wakeup clear register (PWR_WKUPCR).

0: No wakeup event occurred

1: A wakeup event was received from WKUP7 pin

Bit 5 **WKUPF6**: Wakeup pin WKUP6 flag

This bit is set by hardware and cleared only by a reset pin or by setting the WKUPC6 bit in the PWR wakeup clear register (PWR_WKUPCR).

0: No wakeup event occurred

1: A wakeup event was received from WKUP6 pin

Bit 4 **WKUPF5**: Wakeup pin WKUP5 flag

This bit is set by hardware and cleared only by a reset pin or by setting the WKUPC5 bit in the PWR wakeup clear register (PWR_WKUPCR).

0: No wakeup event occurred

1: A wakeup event was received from WKUP5 pin

Bit 3 **WKUPF4**: Wakeup pin WKUP4 flag

This bit is set by hardware and cleared only by a reset pin or by setting the WKUPC4 bit in the PWR wakeup clear register (PWR_WKUPCR).

0: No wakeup event occurred

1: A wakeup event was received from WKUP4 pin

Bit 2 **WKUPF3**: Wakeup pin WKUP3 flag

This bit is set by hardware and cleared only by a reset pin or by setting the WKUPC3 bit in the PWR wakeup clear register (PWR_WKUPCR).

0: No wakeup event occurred

1: A wakeup event was received from WKUP3 pin

Bit 1 **WKUPF2**: Wakeup pin WKUP2 flag

This bit is set by hardware and cleared only by a reset pin or by setting the WKUPC2 bit in the PWR wakeup clear register (PWR_WKUPCR).

0: No wakeup event occurred

1: A wakeup event was received from WKUP2 pin

Bit 0 **WKUPF1**: Wakeup pin WKUP1 flag

This bit is set by hardware and cleared only by a reset pin or by setting the WKUPC1 bit in the PWR wakeup clear register (PWR_WKUPCR).

0: No wakeup event occurred

1: A wakeup event was received from WKUP1 pin

6.8.9 PWR wakeup enable and polarity register (PWR_WKUPEPR)

Address offset: 0x028

Reset value: 0x0000 0000 (reset value after Standby: the value of the register is kept in Standby mode.)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WKUPPUPD8[1:0]		WKUPPUPD7[1:0]		WKUPPUPD6[1:0]		WKUPPUPD5[1:0]		WKUPPUPD4[1:0]		WKUPPUPD3[1:0]		WKUPPUPD2[1:0]		WKUPPUPD1[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WKUP P8	WKUP P7	WKUP P6	WKUP P5	WKUP P4	WKUP P3	WKUP P2	WKUP P1	WKUP EN8	WKUP EN7	WKUP EN6	WKUP EN5	WKUP EN4	WKUP EN3	WKUP EN2	WKUP EN1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **WKUPPUPDn[1:0]**: Wakeup pin pull configuration for WKUPn (n = 8 to 1)

These bits define the I/O pad pull configuration used when WKUPENn = 1. The associated GPIO port pull configuration must be set to the same value or to 00.

The Wakeup pin pull configuration is kept in Standby mode.

00: No pull

01: Pull-up

10: Pull-down

11: Reserved

Bits 15:8 **WKUPPn**: Wakeup pin polarity bit for WKUPn (n = 8 to 1)

These bits define the polarity used for event detection on WKUPn external wakeup pin.

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

Bits 7:0 **WKUPENn**: Enable wakeup pin WKUPn (n = 8 to 1)

Each bit is set and cleared by software.

0: An event on WKUPn pin does not wakeup the system from Standby mode.

1: A rising or falling edge on WKUPn pin wakes-up the system from Standby mode.

Note: Note: An additional wakeup event is detected if WKUPn pin is enabled (by setting the WKUPENn bit) when WKUPn pin level is already high when WKUPPn selects rising edge, or low when WKUPPn selects falling edge.

6.8.10 PWR USB Type-C and power delivery register (PWR_UCPDR)

Address offset: 0x02C

Reset value: 0x0000 0002 (reset value after Standby: the value of the register is kept in Standby mode.)

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

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

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

Bit 1 **UCPD_STBY**: UCPD Standby mode

When set, this bit is used to memorize the UCPD configuration in Standby mode.
This bit must be written to 1 just before entering Standby mode when using UCPD.
It must be written to 0 after exiting the Standby mode and before writing any UCPD registers.

Bit 0 **UCPD_DBDIS**: UCPD dead battery disable

After exiting reset, the USB Type-C “dead battery” behavior is enabled, which may have a pull-down effect on CC1 and CC2 pins. It is recommended to disable it in all cases, either to stop this pull-down or to handover control to the UCPD (the UCPD must be initialized before doing the disable).
0: UCPD dead battery pull-down behavior enabled on UCPDx_CC1 and UCPDx_CC2 pins
1: UCPD dead battery pull-down behavior disabled on UCPDx_CC1 and UCPDx_CC2 pins

6.8.11 PWR apply pull register (PWR_APCR)

Address offset: 0x030

Reset value: 0x0000 0000

(Reset value after Standby: the value of the register is kept in Standby mode.)

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

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

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

Bit 0 **APC**: Apply pull-up and pull-down configuration

0: The I/O pull-up and configurations defined in the I3C_PUCR and XSPI_PUCR/PDCR registers are not applied. The I/Os are set to the default mode in Standby (analog/Hi-Z)
1: The I/O pull-up and pull-down configurations defined in the I3C_PUCR and XSPI_PUCR/PDCR registers are applied in Standby mode and after wakeup until this bit is reset to 0.

6.8.12 PWR I3C pull-up control register (PWR_I3CPUCR)

Address offset: 0x034

Reset value: 0x0000 0000 (reset value after Standby: the value of the register is kept in Standby mode.)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	I3CPI2_PU	I3CPH12_PU	I3CPH11_PU	I3CPG14_PU	I3CPG13_PU	Res.	Res.	I3CPE6_PU	I3CPE5_PU	I3CPD7_PU	I3CPD6_PU	I3CPD1_PU	I3CPD0_PU	I3CPC9_PU
		rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I3CPC8_PU	I3CPB9_PU	I3CPB8_PU	I3CPB4_PU	I3CPB3_PU	I3CPA15_PU	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw										

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **I3CPI2_PU**: Port PI2 I3C pull-up bit

When I3C is configured on PI2, this bit controls the pull-up on I3C1_SCL (PI2) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 28 **I3CPH12_PU**: Port PH12 I3C pull-up bit

When I3C is configured on PH12, this bit controls the pull-up on I3C1_SDA (PH12) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 27 **I3CPH11_PU**: Port PH11 I3C pull-up bit

When I3C is configured on PH11, this bit controls the pull-up on I3C1_SCL (PH11) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 26 **I3CPG14_PU**: Port PG14 I3C pull-up bit.

When I3C is configured on PD6, this bit controls the pull-up on I3C1_SCL (PG14) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 25 **I3CPG13_PU**: Port PG13 I3C pull-up bit

When I3C is configured on PG13, this bit controls the pull-up on I3C1_SDA (PG13) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **I3CPE6_PU**: Port PE6 I3C pull-up bit

When I3C is configured on PE6, this bit controls the pull-up on I3C1_SDA (PE6) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 21 **I3CPE5_PU**: Port PE5 I3C pull-up bit

When I3C is configured on PE5, this bit controls the pull-up on I3C1_SCL (PE5) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 20 **I3CPD7_PU**: Port PD7 I3C pull-up bit

When I3C is configured on PD7, this bit controls the pull-up on I3C1_SDA (PD7) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 19 **I3CPD6_PU**: Port PD6 I3C pull-up bit

When I3C is configured on PD6, this bit controls the pull-up on I3C1_SCL (PD6) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 18 **I3CPD1_PU**: Port PD1 I3C pull-up bit

When I3C is configured on PD1, this bit controls the pull-up on I3C1_SDA (PD1) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 17 **I3CPD0_PU**: Port PD0 I3C pull-up bit

When I3C is configured on PD0, this bit controls the pull-up on I3C1_SCL (PD0) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 16 **I3CPC9_PU**: Port PC9 I3C pull-up bit

When I3C is configured on PC9, this bit controls the pull-up on I3C1_SDA (PC9) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 15 **I3CPC8_PU**: Port PC8 I3C pull-up bit

When I3C is configured on PC8, this bit controls the pull-up on I3C1_SCL (PC8) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 14 **I3CPB9_PU**: Port PB9 I3C pull-up bit

When I3C is configured on PB9, this bit controls the pull-up on I3C1_SDA (PB9) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 13 **I3CPB8_PU**: Port PB8 I3C pull-up bit

When I3C is configured on PB8, this bit controls the pull-up on I3C1_SDA (PB8) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 12 **I3CPB4_PU**: Port PB4 I3C pull-up bit

When I3C is configured on PB4, this bit controls the pull-up on I3C1_SCL (PB4) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 11 **I3CPB3_PU**: Port PB3 I3C pull-up bit

When I3C is configured on PB3, this bit controls the pull-up on I3C1_SDA (PB3) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 10 **I3CPA15_PU**: Port PA15 I3C pull-up bit

When I3C is configured on PA15, this bit controls the pull-up on I3C1_SCL (PA15) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bits 9:0 Reserved, must be kept at reset value.

6.8.13 PWR XSPI pull-up control register (PWR_XSPIPUCR)

Address offset: 0x038

Reset value: 0x0000 0000 (POR and System Reset) (reset value after Standby: the value of the register is kept in Standby mode.)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
XSPIPH0_PU	XSPIPH1_PU	XSPIPF2_PU	XSPIPA0_PU	XSPIPA2_PU	XSPIPA3_PU	XSPIPA4_PU	XSPIPA5_PU	XSPIPB1_PU	XSPIPB2_PU	XSPIPD0_PU	XSPIPD1_PU	XSPIPD2_PU	XSPIPD3_PU	XSPIPG10_PU	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **XSPIPH0_PU**: Port PH0 XSPI pull-up bit

When XSPI is configured on PH0, this bit controls the pull-up on XSPI DQS0 (PH0) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 30 **XSPIPH1_PU**: Port PH1 XSPI pull-up bit

When XSPI is configured on PH1, this bit controls the pull-up on XSPI NCS2 (PH1) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 29 **XSPIPF2_PU**: Port PF2 XSPI pull-up bit

When XSPI is configured on PF2, this bit controls the pull-up on XSPI NCLK (PF2) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 28 **XSPIPA0_PU**: Port PA0 XSPI pull-up bit

When XSPI is configured on PA0, this bit controls the pull-up on XSPI CLK (PA0) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 27 **XSPIPA2_PU**: Port PA2 XSPI pull-up bit

When XSPI is configured on PA2, this bit controls the pull-up on XSPI NCS1 (PA2) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 26 **XSPIPA3_PU**: Port PA3 XSPI pull-up bit

When XSPI is configured on PA3, this bit controls the pull-up on XSPI DQS0 (PA3) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 25 **XSPIPA4_PU**: Port PA4 XSPI pull-up bit

When XSPI is configured on PA4, this bit controls the pull-up on XSPI NCS2 (PA4) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 24 **XSPIPA5_PU**: Port PA5 XSPI pull-up bit

When XSPI is configured on PA5, this bit controls the pull-up on XSPI NCS1 (PA5) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 23 **XSPIPB1_PU**: Port PB1 XSPI pull-up bit

When XSPI is configured on PB1, this bit controls the pull-up on XSPI CLK (PB1) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 22 **XSPIPB2_PU**: Port PB2 XSPI pull-up bit

When XSPI is configured on PB2, this bit controls the pull-up on XSPI NCLK (PB2) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 21 **XSPIPD0_PU**: Port PD0 XSPI pull-up bit

When XSPI is configured on PD0, this bit controls the pull-up on XSPI DQS0 (PD0) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 20 **XSPIPD1_PU**: Port PD1 XSPI pull-up bit

When XSPI is configured on PD1, this bit controls the pull-up on XSPI NCS2 (PD1) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 19 **XSPIPD2_PU**: Port PD2 XSPI pull-up bit

When XSPI is configured on PD2, this bit controls the pull-up on XSPI CLK (PD2) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 18 **XSPIPD3_PU**: Port PD3 XSPI pull-up bit

When XSPI is configured on PD3, this bit controls the pull-up on XSPI NCLK (PD3) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 17 **XSPIPG10_PU**: Port PG10 XSPI pull-up bit

When XSPI is configured on PG10, this bit controls the pull-up on XSPI NCS1 (PG10) in standby mode.

0: Pull-up is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-up is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bits 16:0 Reserved, must be kept at reset value.

6.8.14 PWR XSPI pull-down control register (PWR_XSPIPDCR)

Address offset: 0x03C

Reset value: 0x0000 0000 (reset value after Standby: the value of the register is kept in Standby mode.)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
XSPIPH0_PD	XSPIPH1_PD	XSPIPF2_PD	XSPIPA0_PD	XSPIPA2_PD	XSPIPA3_PD	XSPIPA4_PD	XSPIPA5_PD	XSPIPB1_PD	XSPIPB2_PD	XSPIPD0_PD	XSPIPD1_PD	XSPIPD2_PD	XSPIPD3_PD	XSPIPG10_PD	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **XSPIPH0_PD:** Port PH0 XSPI pull-down bit

When XSPI is configured on PH0, this bit controls the pull-down on XSPI DQS0 (PH0) in standby mode.

0: Pull-down is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-down is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 30 **XSPIPH1_PD:** Port PH1 XSPI pull-down bit

When XSPI is configured on PH1, this bit controls the pull-down on XSPI NCS2 (PH1) in standby mode.

0: Pull-down is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-down is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 29 **XSPIPF2_PD:** Port PF2 XSPI pull-down bit

When XSPI is configured on PF2, this bit controls the pull-down on XSPI NCLK (PF2) in standby mode.

0: Pull-down is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-down is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 28 **XSPIPA0_PD:** Port PA0 XSPI pull-down bit

When XSPI is configured on PA0, this bit controls the pull-down on XSPI CLK (PA0) in standby mode.

0: Pull-down is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-down is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 27 **XSPIPA2_PD:** Port PA2 XSPI pull-down bit

When XSPI is configured on PA2, this bit controls the pull-down on XSPI NCS1 (PA2) in standby mode.

0: Pull-down is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-down is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 26 **XSPIPA3_PD:** Port PA3 XSPI pull-down bit

When XSPI is configured on PA3, this bit controls the pull-down on XSPI DQS0 (PA3) in standby mode.

0: Pull-down is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-down is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 25 **XSPIPA4_PD:** Port PA4 XSPI pull-down bit

When XSPI is configured on PA4, this bit controls the pull-down on XSPI NCS2 (PA4) in standby mode.

0: Pull-down is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-down is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 24 **XSPIPA5_PD:** Port PA5 XSPI pull-down bit

When XSPI is configured on PA5, this bit controls the pull-down on XSPI NCS1 (PA5) in standby mode.

0: Pull-down is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-down is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 23 **XSPIPB1_PD:** Port PB1 XSPI pull-down bit

When XSPI is configured on PB1, this bit controls the pull-down on XSPI CLK (PB1) in standby mode.

0: Pull-down is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-down is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 22 **XSPIPB2_PD:** Port PB2 XSPI pull-down bit

When XSPI is configured on PB2, this bit controls the pull-down on XSPI NCLK (PB2) in standby mode.

0: Pull-down is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-down is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 21 **XSPIPD0_PD:** Port PD0 XSPI pull-down bit

When XSPI is configured on PD0, this bit controls the pull-down on XSPI DQS0 (PD0) in standby mode.

0: Pull-down is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-down is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 20 **XSPIPD1_PD:** Port PD1 XSPI pull-down bit

When XSPI is configured on PD1, this bit controls the pull-down on XSPI NCS2 (PD1) in standby mode.

0: Pull-down is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-down is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 19 **XSPIPD2_PD:** Port PD2 XSPI pull-down bit

When XSPI is configured on PD2, this bit controls the pull-down on XSPI CLK (PD2) in standby mode.

0: Pull-down is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-down is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 18 **XSPIPD3_PD:** Port PD3 XSPI pull-down bit

When XSPI is configured on PD3, this bit controls the pull-down on XSPI NCLK (PD3) in standby mode.

0: Pull-down is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-down is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bit 17 **XSPIPG10_PD:** Port PG10 XSPI pull-down bit

When XSPI is configured on PG10, this bit controls the pull-down on XSPI NCS1(PG10 in standby mode.

0: Pull-down is disabled in standby mode

1: If PWR_APCR. APC = 1, pull-down is active in standby mode and after wakeup until PWR_APCR. APC is reset to 0.

Bits 16:0 Reserved, must be kept at reset value.

6.8.15 PWR register map

Table 48. Power control register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PWR_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALS[1:0]		AVDEN	AVDREADY	BOOSTE		FLPS	DBP	PLS[2:0]			PVDE				SVOS
	Reset value																	0	0	0	0	0		0	0	0	0	0	0				1
0x004	PWR_SR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		AVDO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	ACTVOSRDY	ACTVOS[1:0]	
	Reset value																			0									0		0	1	0
0x008	PWR_CSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEMPH	TEMPL	VBATH	VBATL	Res.	Res.	Res.	BRRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MONEN	Res.	Res.	Res.	BREN
	Reset value									0	0	0	0				0												0				0
0x00C	PWR_CSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDRDY	Res.	Res.	Res.	Res.	Res.	Res.	VBRS	VBE	Res.	Res.	Res.	Res.	Res.	SMPSEN	LDOEN	BYPASS
	Reset value																0							0	0						1	1	0
0x010	PWR_CSR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SBF	STOPF	Res.	Res.	Res.	Res.	Res.	Res.	CSSF	PDDS
	Reset value																							0	0							0	0
0x014	PWR_CSR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VOSRDY	VOS[1:0]	
	Reset value																														1	1	0
0x020	PWR_WKUPCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPC8	WKUPC7	WKUPC6	WKUPC5	WKUPC4	WKUPC3	WKUPC2	WKUPC1
	Reset value																									0	0	0	0	0	0	0	0
0x024	PWR_WKUPFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF8	WKUPF7	WKUPF6	WKUPF5	WKUPF4	WKUPF3	WKUPF2	WKUPF1
	Reset value																									0	0	0	0	0	0	0	0

Table 48. Power control register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x028	PWR_WKUPEPR	WKUPPUPD8[1:0]		WKUPPUPD7[1:0]		WKUPPUPD6[1:0]		WKUPPUPD5[1:0]		WKUPPUPD4[1:0]		WKUPPUPD3[1:0]		WKUPPUPD2[1:0]		WKUPPUPD1[1:0]		WKUPP8	WKUPP7	WKUPP6	WKUPP5	WKUPP4	WKUPP3	WKUPP2	WKUPP1	WKUPEN8	WKUPEN7	WKUPEN6	WKUPEN5	WKUPEN4	WKUPEN4	WKUPEN2	WKUPEN1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	PWR_UCPDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD_STDBY	UCPD_DBDIS
	Reset value																															1	0
0x030	PWR_APCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APC
	Reset value																																0
0x034	PWR_I3CPUCR	Res.	Res.	I3CPI2_PU	I3CPH12_PU	I3CPH11_PU	I3CPG14_PU	I3CPG13_PU	Res.	Res.	I3CPE6_PU	I3CPE5_PU	I3CPD7_PU	I3CPD6_PU	I3CPD1_PU	I3CPD0_PU	I3CPC9_PU	I3CPC8_PU	I3CPB9_PU	I3CPB8_PU	I3CPB4_PU	I3CPB3_PU	I3CPA15_PU	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0										
0x038	PWR_XSPIPUCR	XSPIPH0_PU	XSPIPH1_PU	XSPIPF2_PU	XSPIPA0_PU	XSPIPA2_PU	XSPIPA3_PU	XSPIPA4_PU	XSPIPA5_PU	XSPIPB1_PU	XSPIPB2_PU	XSPIPD0_PU	XSPIPD1_PU	XSPIPD2_PU	XSPIPD3_PU	XSPIPG10_PU	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																	
0x03C	PWR_XSPIPDCR	XSPIPH0_PD	XSPIPH1_PD	XSPIPF2_PD	XSPIPA0_PD	XSPIPA2_PD	XSPIPA3_PD	XSPIPA4_PD	XSPIPA5_PD	XSPIPB1_PD	XSPIPB2_PD	XSPIPD0_PD	XSPIPD1_PD	XSPIPD2_PD	XSPIPD3_PD	XSPIPG10_PD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																	

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

7 Reset and clock control (RCC)

The RCC block manages the clock and reset generation for the whole microcontroller.

The operating modes to which this section refers are defined in *Section 6.6.1: Operating modes* of the PWR block.

7.1 RCC main features

AHB-Lite bus interface

Reset block:

- Generation of local and system reset
- Bidirectional pin reset allowing to reset the microcontroller or external devices
- WWDG and IWDG reset supported
- Power-on (POR) and Brownout (BOR) resets initiated by the power control (PWR)

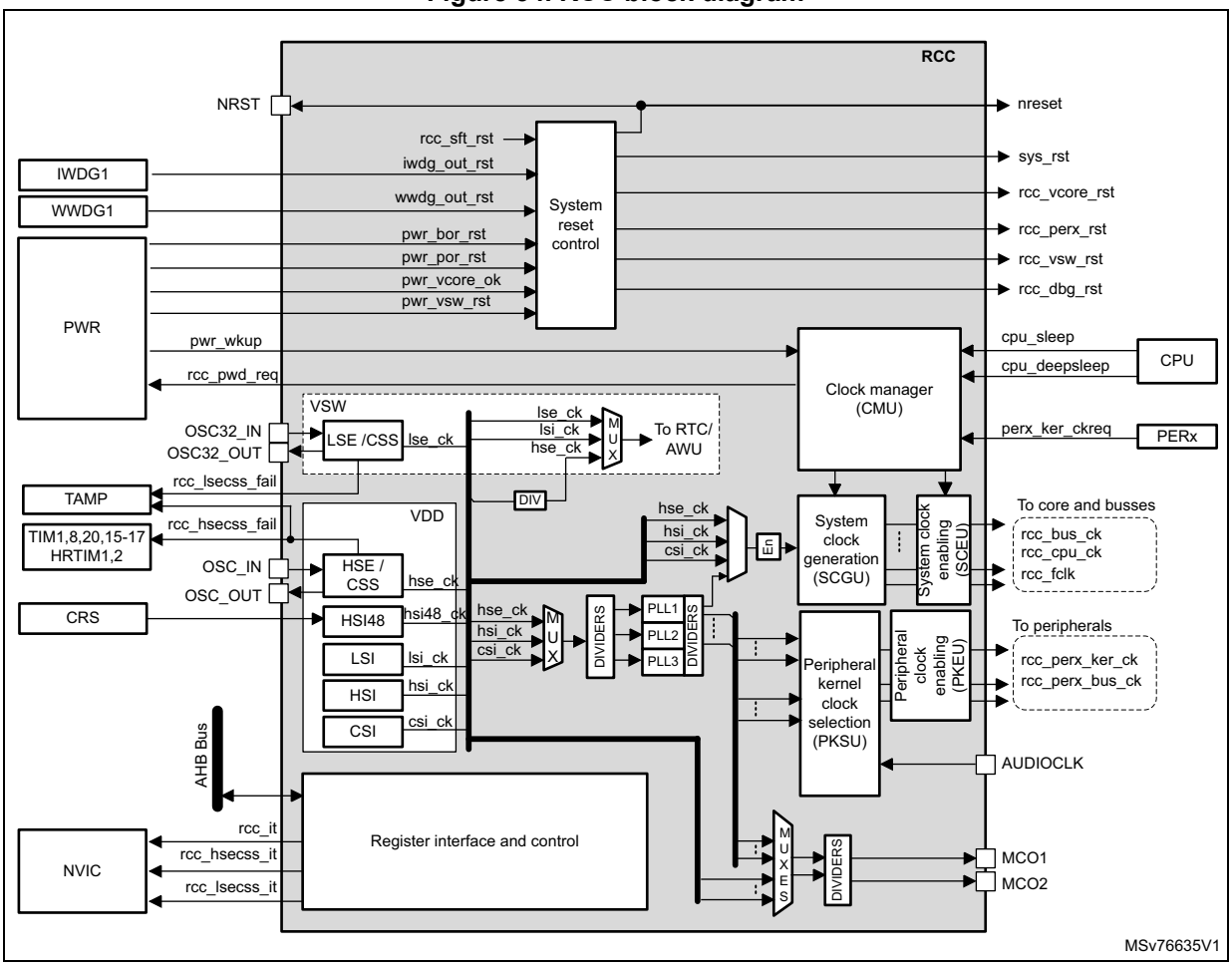
Clock generation block:

- Generation and dispatching of clocks for the complete device
- 3 separate PLLs using integer or fractional ratios
- Possibility of changing the PLL fractional ratios on-the-fly
- Smart clock gating to reduce power dissipation
- System clock protection
- 2 external oscillators:
  - High-speed external oscillator (HSE) supporting a wide range of crystals from 4 to 50 MHz frequency
  - Low-speed external oscillator (LSE) for the 32 kHz crystals
- 4 internal oscillators
  - High-speed internal oscillator (HSI)
  - 48 MHz RC oscillator (HSI48)
  - Low-power internal oscillator (CSI)
  - Low-speed internal oscillator (LSI)
- Buffered clock outputs for external devices
- Generation of two types of interrupts lines:
  - Dedicated interrupt lines for clock security management
  - One general interrupt line for other events
- Clock generation handling in Stop and Standby mode

7.2 RCC block diagram

Figure 34 shows the RCC block diagram.

Figure 34. RCC block diagram



7.3 RCC pins and internal signals

Table 49 lists the RCC inputs and output signals connected to package pins or balls.

Table 49. RCC input/output signals connected to package pins or balls

Signal name	Signal type	Description
NRST	I/O	System reset, can be used to provide reset to external devices
OSC32_IN	I	32 kHz oscillator input
OSC32_OUT	O	32 kHz oscillator output
OSC_IN	I	System oscillator input
OSC_OUT	O	System oscillator output
MCO1	O	Output clock 1 for external devices
MCO2	O	Output clock 2 for external devices
AUDIOCLK	I	External kernel clock input for digital audio interfaces: SPI/I2S and ADF

The RCC exchanges a lot of internal signals with all components of the product. For that reason, *Table 50* only shows the most significant internal signals.

Table 50. RCC internal input/output signals

Signal name	Signal type	Description
rcc_it	O	General interrupt request line
rcc_hsecss_it	O	HSE clock security failure interrupt
rcc_lsecss_it	O	LSE clock security failure interrupt
rcc_hsecss_fail	O	Event indicating that a HSE clock security failure is detected. This signal is connected to TIMERS.
rcc_lsecss_fail	O	Event indicating that a LSE clock security failure is detected. This signal is connected to TAMP, EXTI and PWR
nreset	I/O	Application reset
sys_rst	I/O	System reset
iwdg_out_rst	I	Reset line driven by the IWDG, indicating that a timeout occurred
wwdg_out_rst	I	Reset line driven by the WWDG, indicating that a timeout occurred
pwr_bor_rst	I	Brownout reset generated by the PWR block
pwr_por_rst	I	Power-on reset generated by the PWR block
pwr_vsw_rst	I	Power-on reset of the V_{SW} domain generated by the PWR block
rcc_perx_rst	O	Reset generated by the RCC for the peripherals
pwr_wkup	I	Wakeup domain request generated by the PWR and used to restore the domain clocks
rcc_pwd_req	O	Low-power request generated by the RCC and used to set the circuit into low-power mode

Table 50. RCC internal input/output signals (continued)

Signal name	Signal type	Description
cpu_sleep	I	Signals generated by the CPU, indicating if the CPU is in CRun, Sleep or CStop
cpu_deepsleep	I	
perx_ker_ckreq	I	Signal generated by some peripherals in order to request the activation of their kernel clock
rcc_perx_ker_ck	O	Kernel clock signals generated by the RCC, for some peripherals
rcc_perx_bus_ck	O	Bus interface clock signals generated by the RCC for peripherals
rcc_bus_ck	O	Clocks for APB, AHB and AXI bridges generated by the RCC
rcc_cpu_ck	O	Clocks for the CPU, generated by the RCC
rcc_fclk	O	

7.4 RCC reset block functional description

The RCC handles the reset generation for the complete product, using events coming from different sources:

- Assertion of the NRST pin from an external device
- A failure on the supply voltage applied to VDD, VBAT
- An exit from Standby
- A watchdog timeout
- A software command

The reset scope depends on the source that generates the reset.

7.4.1 Reset from PWR block

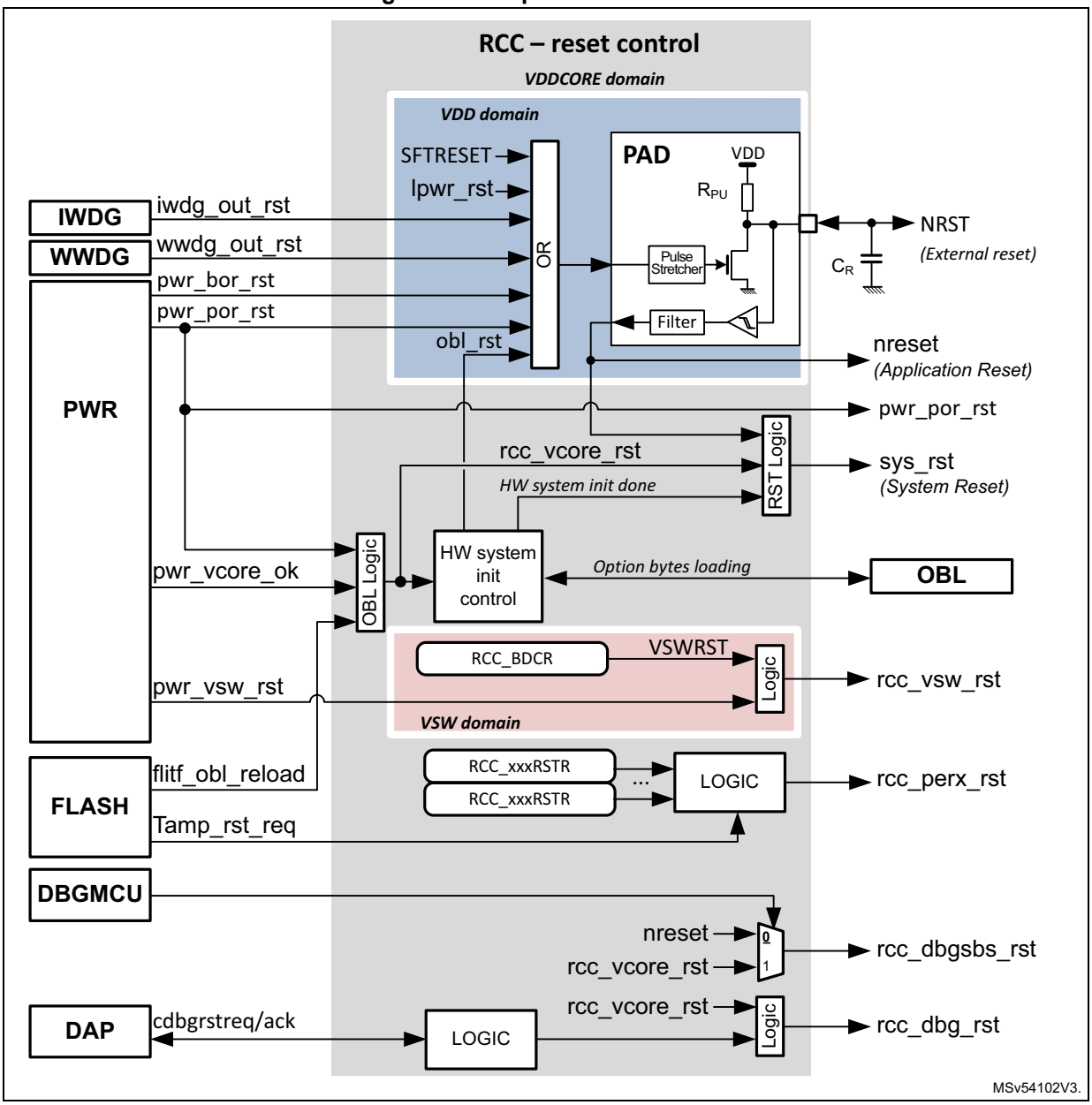
The PWR block provides several reset signals to the RCC:

- The power-on/off reset signal (**pwr_por_rst**), which is asserted when the VDD supply is lower than the VPOR threshold.
- The brown-out reset signal (**pwr_bor_rst**), which is asserted when the VDD supply is lower than the VBOR threshold.
- The core reset signal (**pwr_vcore_ok**), which is asserted when the VDDCORE supply is not valid or available. Note that the V_{DDCORE} voltage is switched off when the product is in Standby. So when the system exits from Standby mode, the **pwr_vcore_ok** signal is asserted as long the V_{DDCORE} voltage provided by the internal regulator is not valid. Note that when the V_{DD} supply is not valid, **pwr_vcore_ok** is asserted as well.
- The V_{SW} domain reset signal (**pwr_vsw_rst**), which is asserted when the VSW supply is lower than the expected threshold.

*Note: The **rcc_vcore_rst** is generated from the **pwr_vcore_ok** signal, the main difference between these two signals is that **rcc_vcore_rst** remains asserted until the option byte loading operation is completed.*

Refer to *Table 51: Reset coverage summary* for details.

Figure 35. Simplified reset circuit



7.4.2 The system and application resets (sys_rst and nreset)

A system reset (**sys_rst)** resets most of the registers to their default values unless otherwise specified in the register description.

A system reset can be generated from one of the following sources:

- An assertion of the NRST pin (external reset)
- A reset from the power-on/off reset block (**pwr_por_rst**)
- A reset from the brownout reset block (**pwr_bor_rst**)

  Refer to *Section 6.5.2: Brownout reset (BOR)* for a detailed description of the BOR function.
- A reset from the independent watchdogs (**iwdg_out_rst**)
- An exit from Standby (**rcc_vcore_rst**).
- A reset from the window watchdogs depending on WWDG configuration (**wwdg_out_rst**)
- A software reset from the Cortex®-M7 core

  It is generated via the SYSRESETREQ signal issued by the Cortex®-M7 core. This signal is also named SFTRESET in this document.
- A reset from the low-power mode security reset, depending on option byte configuration (**lpwr_rst**).
- An option byte reload request from the flash interface (**obl_rst**)

The application reset (**nreset**) is similar to the system reset, but it is not asserted when the system exits from Standby.

Note: The SYSRESETREQ bit in Cortex®-M7 must be set through the FPU application interrupt and reset control register to force a software reset on the device. Refer to the Cortex®-M7 with FPU technical reference manual for more details (see http://infocenter.arm.com).

7.4.3 The NRST reset

The NRST is active low. A pulse stretcher guarantees a minimum reset pulse duration of 20 µs (Refer to datasheet for details). In addition it is possible to extend the NRST assertion thanks to C_R capacitor. It is not recommended to let the NRST pin unconnected.

When it is not used, connect this pin to the ground via a 10 to 100 nF capacitor (C_R in *Figure 35*). As shown in *Figure 35*, a filter is also present in order to suppress spurs coming from NRST PAD.

7.4.4 Low-power mode security reset (lpwr_rst)

To prevent critical applications from mistakenly enter a low-power mode, two low-power mode security resets are available. When enabled through nRST_STOP and nRST_STANDBY option bytes, a system reset is generated if the following conditions are met:

- The CPU accidentally enters Stop mode.
  This type of reset is enabled by resetting nRST_STOP user option byte. In this case, whenever the Stop mode entry sequence is successfully executed, a system reset is generated.
- The system accidentally enters Standby mode.
  This type of reset is enabled by resetting nRST_STANDBY user option byte. In this

case, whenever a Standby mode entry sequence is successfully executed, a system reset is generated.
When the Standby mode is entered, a flag is also set in the power controller.

LPWRRSTF bit in the *RCC Reset status register (RCC_RSR)* indicates that a low-power mode security reset occurred (see line #7 in *Table 52*).

lpwr_rst is activated when a low-power mode security reset due to CPU occurred.

Refer to *Section 5.4: FLASH option bytes* for additional information.

Refer to *Section 6: Power control (PWR)* for additional information and *Table 40: Operating mode summary* for the overview of the existing power modes.

7.4.5 Backup domain reset

A backup domain reset (**pwr_vsw_rst**) is generated when one of the following events occurs:

- Software reset (triggered by setting the VSWRST bit in the *RCC Backup domain control register (RCC_BDCR)*). All RTC registers and the RCC_BDCR register are set to their reset values, with the exception of LSEDRV, which is not changed if LSEON=1. The BKPSRAM is not affected.
- When the V_{SW} voltage is outside the operating range. All RTC registers and the RCC_BDCR register are set to their reset values. In this case the content of the BKPSRAM is no longer valid.

Refer to *Section 6.4.3: Backup domain* section of PWR block and to section “System security” for additional information.

7.4.6 Coresight debug reset

The coresight debug components can be reset in different ways:

- Over the DAP by setting to 1 the bit CDBGRSTREQ of the control/status register of the debug port. This action asserts the debug reset request signal cdbgrstreq connected to the RCC. Then the RCC activates the debug reset (**rcc_dbg_rst**), a handshake signal cdbgrstack is provided to the DAP request. The debug reset remains as long as the cdbgrstreq is to 1, refer to *Figure 35* for details.
- When V_{DDCORE} power-on reset occurs (**rcc_vcore_rst**). This reset is activated after a power-on reset, or when the product exits from Standby.

Refer to Section 1.7.1: CoreSight Debug Reset for details(TBD xref to Debug section?).

7.4.7 Option bytes loading

As shown in *Figure 35*, the option bytes loading (OBL) sequence is triggered under the following conditions:

- After a power-on, or
- When the system exits from Standby
- When the flash interface requested a reload of option bytes

The system reset (**sys_rst**) can be released only when the option byte loading (OBL) is completed.

7.4.8 Peripheral resets

The application can individually reset any peripheral whenever requested. This can be done via the RCC_xxxxRSTR registers where “xxxx” is the name of the bus to which the peripheral is connected.

In order to reset a peripheral, the corresponding reset bit must be set to 1, and then set back to 0. There is no need to enable the peripheral clock in order to reset a peripheral.

Note also that the CRYP, SAES and HASH blocks can be reset in the case of a tamper event.

7.4.9 Reset coverage summary

Table 51: Reset coverage summary gives a detailed view of the reset coverage of the most important reset sources.

Note: When V_{DD} is not valid, V_{DDCORE} is also not valid.

Table 51. Reset coverage summary(1)

Functions that are reset		Main reset lines						
		pwr_por_rst (2)	rcc_vcore_rst	sys_rst	nreset (4)	rcc_dbg_rst	rcc_perx_rst	rcc_vsw_rst
V_{DD} domain		x	-	-	-	-	-	-
MCU		x	x	x	-	-	-	-
WWDG		x	x	x	-	-	-	-
IWDG		x	-	-	x	-	-	-
AXI/AHB Interconnections.		x	x	x	-	-	-	-
Debug components, including DBGMCU Reset all the debug parts except the SWJ-DP function. The SWJ-DP is reset by the NJTRST or rcc_vcore_rst resets. The bit DCRT on DBGMCU_CR is reset with pwr_por_rst.		x	x	-	-	x	-	-
Hardware system init. It includes the reload of the option bytes.		x	x	-	-	-	-	-
RCC	*RCC Reset status register (RCC_RSR)*	x	-	-	-	-	-	-
	RCC clock control and status register (RCC_CSR)	x	-	-	-	-	-	-
	RCC Backup domain control register (RCC_BDCR)	-	-	-	-	-	-	x
	Other RCC registers	x	x	x	-	-	-	-
PWR	PWR_CSR1	-	-	-	-	-	-	x
	PWR_CSR2	x	-	-	-	-	-	-
	PWR_CSR3 Individual bits of this register do not have the same reset condition, refer to the PWR section for details.	x	x	x	-	-	-	-
	PWR_WKUPCR, PWR_WKUPFR, PWR_WKUPEPR	x	-	-	x	-	-	-
	Other registers	x	x	x	-	-	-	-
RTC		-	-	-	-	-	-	x
BKPSRAM After a reset of the V_{SW} domain, the backup regulator of the BKPSRAM is disabled, this function is controlled via the PWR block, BREN bit. If the rcc_vsw_rst reset is due to a too low V_{SW} voltage, the content of BKPSRAM is lost.		-	-	-	-	-	-	x
SBS	SBS_RSSCMDR	x	-	-	-	-	-	-
Other Peripherals		x	x	x	-	-	x	-

1. The ‘x’ means that the function is reset by the corresponding reset line, the ‘-’ means that the function is not reset by the corresponding reset line.
2. The pwr_por_rst is asserted when the voltage applied to VDD is not valid.
When pwr_por_rst is asserted, the rcc_vcore_rst, NRST, the sys_rst and the nreset, are asserted as well.

3. The rcc_vcore_rst is asserted when the voltage applied to VDD is not valid, or when the system exits from Standby (because V_{DDCORE} was switched off).
   When rcc_vcore_rst is asserted, the sys_rst, and the pwr_dbg_rst are asserted as well.
4. When nreset is asserted, the sys_rst is also asserted.

7.4.10 Reset source identification

The CPU can identify the reset source by checking the reset flags in the RCC_RSR register.

The CPU can reset the flags by setting the RMVF bit.

Table 52 shows how the status bits of the RCC_RSR register behave according to the situation that generated the reset. For example, when an IWDG timeout occurs (line #6), if the CPU is reading the RCC_RSR register during the boot phase, both PINRSTF and IWDGRSTF bits are set, indicating that the IWDG also generated a pin reset.

Table 52. Reset source identification (RCC_RSR)(1)

#	Situations generating a reset	SBF (2)	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	BORRSTF	OBLRSTF
1	Power-on reset (**pwr_por_rst**)	0	0	0	0	0	1	1	1	0
2	Pin reset (NRST)	0	0	0	0	0	0	1	0	0
3	Brownout reset (**pwr_bor_rst**)	0	0	0	0	0	0	1	1	0
4	System reset generated by CPU (SFTRESET)	0	0	0	0	1	0	1	0	0
5	WWDG reset (**wwdg_out_rst**)	0	0	1	0	0	0	1	0	0
6	IWDG reset (**iwdg_out_rst**)	0	0	0	1	0	0	1	0	0
7	CPU erroneously enters Stop or Standby mode	0	1	0	0	0	0	1	0	0
8	The flash interface requested a reload of option bytes	0	0	0	0	0	0	1	0	1
9	The product exits from Standby	1	0	0	0	0	0	1	0	0

1. Grayed cells highlight the register bits that are set.
2. The SBF bit is located into the PWR_CSR3 register of the PWR block

7.4.11 Power-on and wakeup sequences

For detailed diagrams refer to *Section 6.4.1: System supply startup* in the PWR section.

The time interval between the event that exits the product from low power and the moment where the CPU is able to execute code, depends on the system state and on its configuration. *Figure 36* shows the most usual examples.

Power-on wakeup sequence

The power-on wakeup sequence shown in *Figure 36* gives the most significant phases of the power-on sequence. It is the longest sequence since the circuit was not powered. Note that this sequence remains unchanged, whatever V_{BAT} was present or not.

Boot from pin reset

When a pin reset occurs, V_{DD} is still present. As a result:

- The regulator settling time is faster since the reference voltage is already stable.
- The HSI restart delay may be needed if the HSI was not enabled when the NRST occurred, otherwise this restart delay phase is skipped.
- The flash memory power recovery delay can also be skipped if the flash memory was enabled when the NRST occurred.

Note: The boot sequence is similar for pwr_bor_rst, lpwr_rst, STFRESET, iwdg_out_rst and wwdg_out_rst.

Boot from system Standby

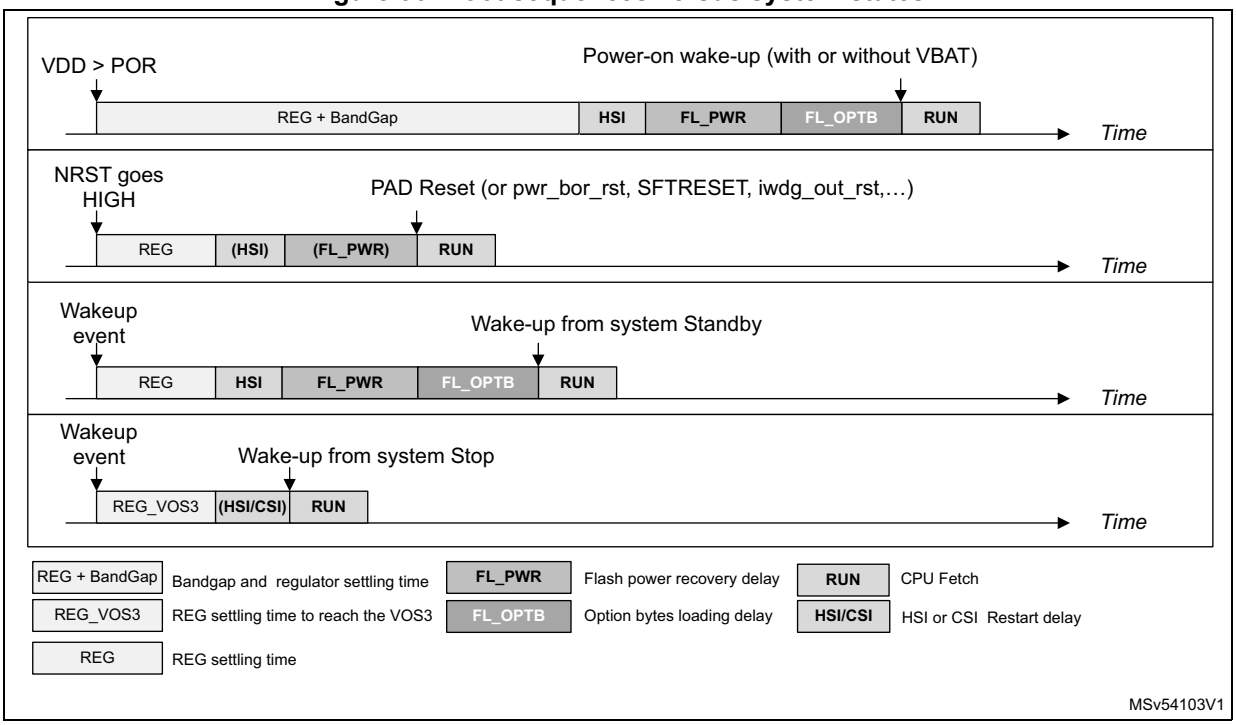
When waking up from system Standby, the reference voltage is stable since V_{DD} has not been removed. As a result, the regulator settling time is fast. Since V_{CORE} was not present, the restart delay for the HSI, the flash memory power recovery and the option byte reloading cannot be skipped.

Restart from system Stop

When restarting from Stop, V_{DD} is still present. As a result, the sequence is mainly composed of two steps:

1. Regulator settling time to reach VOS LOW (default voltage).
2. HSI/CSI restart delay. This step can be skipped if the HSIKERON or CSIKERON bit is set to 1 in the *RCC register map*.

Figure 36. Boot sequences versus system states



7.5 RCC clock block functional description

The RCC provides a wide choice of clock generators:

- HSI (high-speed internal oscillator) clock: ~8, 16, 32 or 64 MHz
- HSE (high-speed external oscillator) clock: 4 to 50 MHz
- LSE (low-speed external oscillator) clock: 32 kHz
- LSI (low-speed internal oscillator) clock: ~32 kHz
- CSI (low-power internal oscillator) clock: ~4 MHz
- HSI48 (high-speed internal oscillator) clock: ~48 MHz

The RCC offers a high flexibility for the application to select the appropriate clock for CPU and peripherals, in particular for peripherals that require a specific clock such as SPI/I2S and ADF.

To optimize the power consumption, each clock source can be switched ON or OFF independently.

The RCC provides up to 3 PLLs; each of them can be configured in integer, with or without SSCG, or fractional mode.

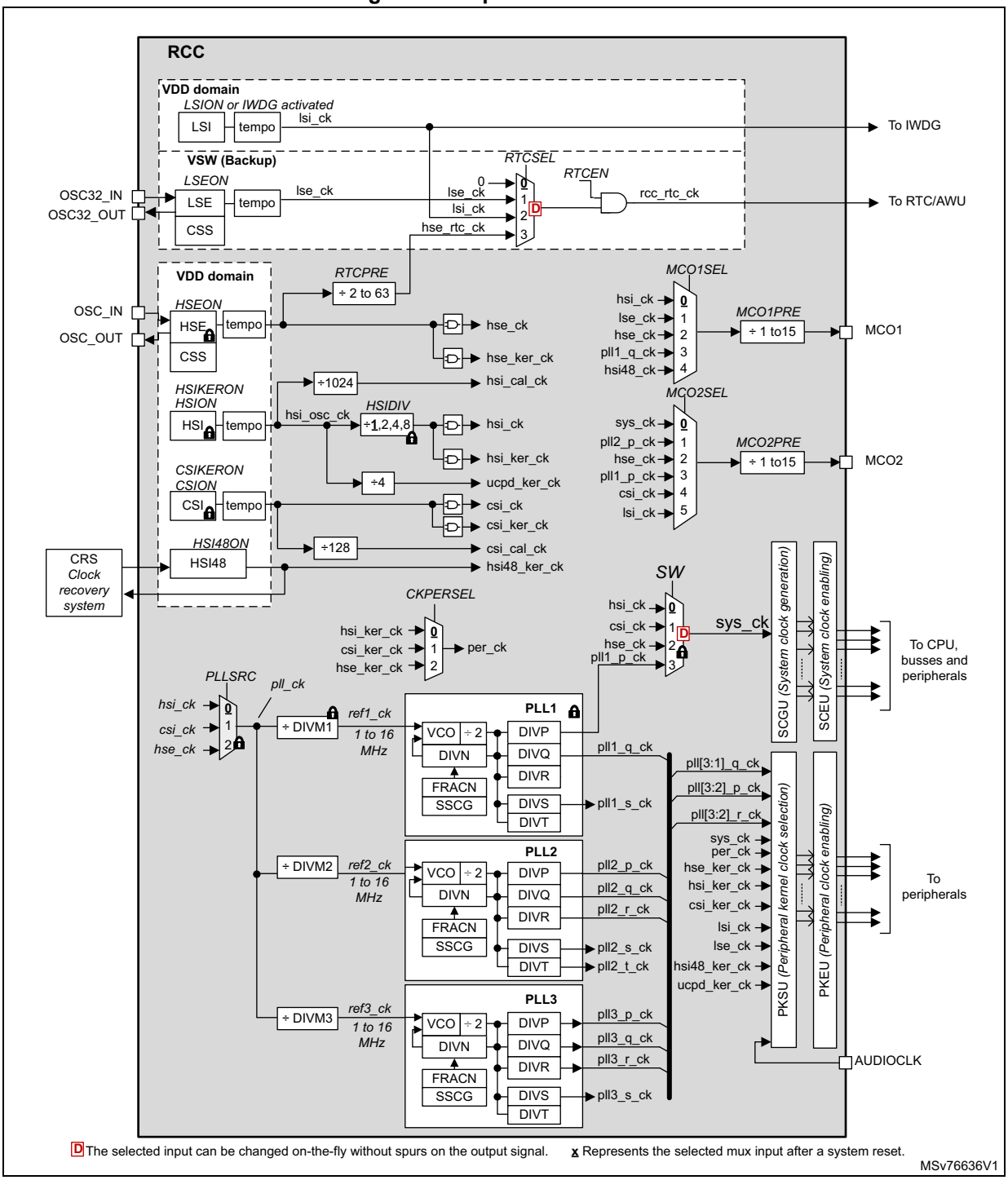
As shown in the *Figure 37*, the RCC offers two clock outputs (MCO1 and MCO2), with a great flexibility in the clock selection and frequency adjustment.

The SCGU block (system clock generation unit) contains several prescalers used to configure the CPU and bus matrix clock frequencies.

The PKSU block (peripheral kernel clock selection unit) provides several dynamic switches allowing a large choice of kernel clock distribution to peripherals.

The PKEU (peripheral kernel clock enable unit) and SCEU (system clock enable unit) blocks perform the peripheral kernel clock gating, and the bus interface/cores/bus matrix clock gating, respectively.

Figure 37. Top-level clock tree



7.5.1 Clock naming convention

The RCC provides clocks to the complete circuit. To avoid misunderstanding, the following terms are used in this document:

- **Peripheral clocks**

  The peripheral clocks are the clocks provided by the RCC to the peripherals. Two kinds of clock are available:

  – Bus interface clocks

  – Kernel clocks

  A peripheral receives from the RCC a bus interface clock in order to access its registers, and thus control the peripheral operation. This clock is generally the AHB, APB or AXI clock depending on which bus the peripheral is connected to. Some peripherals only need a bus interface clock (such as RNG, TIMx).

  Some peripherals also require a dedicated clock to handle the interface function. This clock is named kernel clock. As an example, peripherals such as I2S must generate specific and accurate master clock frequencies, which require dedicated kernel clock frequencies. Another advantage of decoupling the bus interface clock from the specific interface needs, is that the bus clock can be changed without reprogramming the peripheral.

- **CPU clock**

  The CPU clock is the clock provided to the CPU. It is derived from the system clock (**sys_ck**).

- **Bus matrix clocks**

  The bus matrix clocks are the clocks provided to the different bridges (APB, AHB or AXI). These clocks are derived from the system clock (**sys_ck**).

7.5.2 Oscillators description

The table hereafter shows the oscillator states versus system modes, when the oscillators are enabled via registers. The term “available” means that the resource can be used if activated via registers.

Table 53. Oscillator states versus system modes

System modes	V_{DD} domain					V_{SW} domain
	HSE	HSI	HSI48	CSI	LSI	LSE
Exit from system reset	OFF	ON	OFF	OFF	Available	Available
Exit from system Stop	OFF	ON (1)	OFF	ON (2)	Available	Available
In Run/Sleep	Available	Available	Available	Available	Available	Available
In Stop	OFF	Available (3)	OFF	Available (4)	Available	Available
In Standby	OFF	OFF	OFF	OFF	Available	Available
In VBAT	OFF	OFF	OFF	OFF	OFF	Available

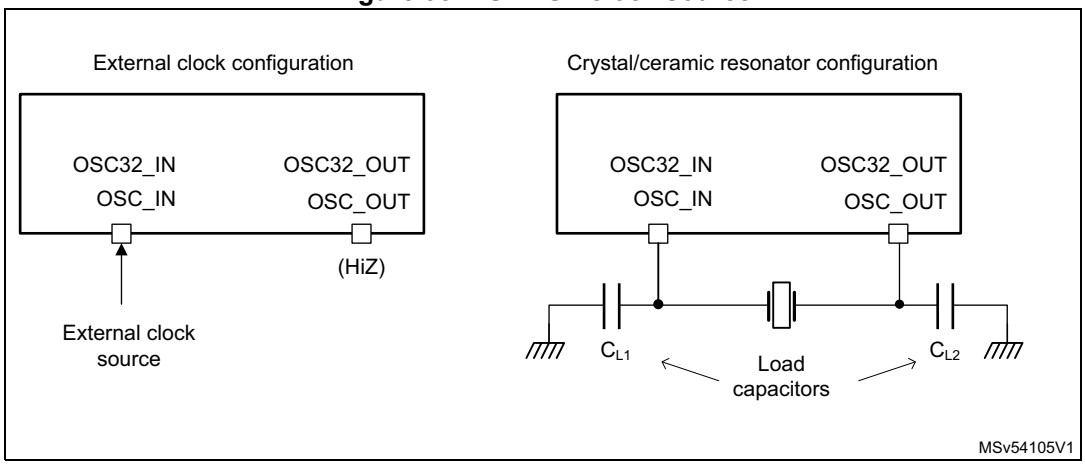
1. If STOPWUCK = 0 or STOPKERWUCK = 0.
2. If STOPWUCK = 1 or STOPKERWUCK = 1.
3. HSI can remain activated in Stop if HSIKERON = 1, or if a peripheral selecting HSI, generates a kernel clock request.
4. CSI can remain activated in Stop if CSIKERON = 1, or if the peripheral selecting CSI, generates a kernel clock request.

HSE oscillator

The HSE block allows the application to provide a very accurate high-speed clock for the product. The HSE can generate an internal clock from two possible sources:

- External clock source
- External crystal/ceramic resonator

Figure 38. HSE/LSE clock source



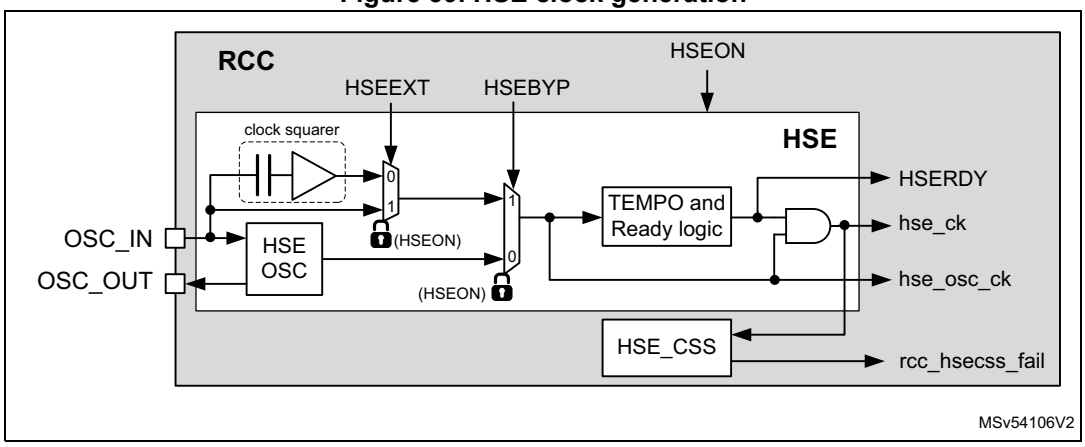
External clock source (HSE bypass)

In this mode the oscillator is not used, and an external clock source must be provided to OSC_IN pin. The external clock can be low swing (analog) or digital. The OSC_OUT pin must be left high-Z (see *Figure 38*).

The external clock signal can be digital or analog (square, sinus or triangle). An analog clock signal with a reduced amplitude is supported thanks to an internal clock squarer. The input signal must have a duty cycle close to 50%. Refer to the datasheet for additional information.

This mode is selected when HSEBYP = 1, and HSEON = 1. In case of an analog clock input (low swing) the HSEEXT must be set to 0, for a digital clock input, HSEEXT bits must be set to 1.

Figure 39. HSE clock generation



External crystal/ceramic resonator

A crystal/resonator can be connected as shown in *Figure 38*: the crystal/resonator and the load capacitors must be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected crystal or ceramic resonator. Refer to the electrical characteristics section of the datasheet for more details.

The oscillator mode is enabled by setting the HSEBYP bit to 0 and HSEON bit to 1.

HSE ready logic

The HSERDY flag indicates when a valid clock is available at HSE output (**hse_ck**). When the HSE is enabled (HSEON set to '1'), the HSERDY flag goes to 1 when 256 valid cycles of HSE have been detected. The **hse_ck** clock is not released until HSERDY goes to 1.

An interrupt can be generated if enabled in the *RCC clock source interrupt enable register (RCC_CIER)*.

HSE controls

The HSE can be switched ON and OFF through the HSEON bit.

The hardware does not allow modifying HSEON bit if one of the following conditions is met:

- The HSE is used directly (via SW switch) as system clock.
- The HSE is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via SW switch).
- The bit XSPICKP is set to 1.
- The bit FMCCKP is set to 1.

The hardware does not allow changing the values of HSEBYP and HSEEXT when HSEON = 1. Bits HSEBYP, HSEEXT and HSEON are located into the *RCC register map*.

The HSE is automatically disabled by hardware when the system enters Stop or Standby mode (See *Table 53: Oscillator states versus system modes*).

In addition, the HSE clock can be driven to the MCO1 and MCO2 outputs and used as clock source for other application components.

Programming sequence

In order to initialize the HSE, the application must follow the sequence hereafter:

- Make sure the HSE is not directly or indirectly used as system clock, if it is the case, use HSI or CSI as clock source for system clock.
- Make sure that XSPICKP and FMCCKP are set to 0
- Disable the HSE by writing to 0 the HSEON bit,
- Check that the HSE is disabled by waiting that the bit HSERDY is set to 0,
- If the oscillator mode is needed:
  - Select the oscillator mode by setting HSEBYP to 0,
- If an external clock is connected to OSC_IN:
  - Select the bypass mode by setting HSEBYP to 1,
  - If the input clock is a full-swing digital signal, set HSEEXT to 1
  - If the input clock is a low-swing signal, set HSEEXT to 0
- Enable again the HSE by writing the HSEON bit to 1,
- Wait for HSERDY = 1, then the HSE is ready for use.

LSE oscillator

The LSE block allows the application to provide a very accurate low-frequency clock for the product. The LSE can generate an internal clock from two possible sources:

- External user clock
- External crystal/ceramic resonator

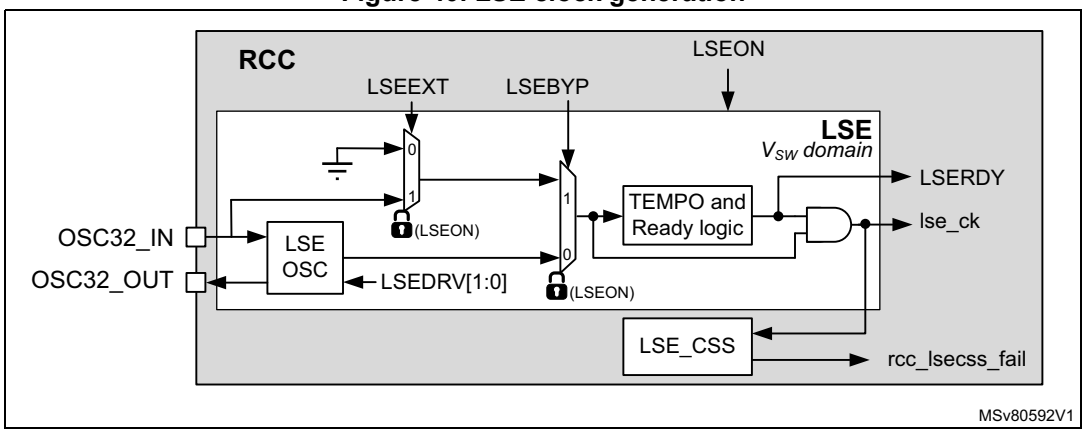
External clock source (LSE bypass)

In this mode, the oscillator is not used, and an external clock source must be provided to OSC32_IN pin. The OSC32_OUT pin must be left high-Z (see *Figure 38*).

The external clock signal can have a frequency up to 1 MHz. The signal must comply with the GPIO digital input requirements.

This mode is selected by setting the LSEBYP, LSEEXT, and LSEON bits to 1.

Figure 40. LSE clock generation



External crystal/ceramic resonator source (LSE crystal)

The LSE clock is generated from a 32.768 kHz crystal or ceramic resonator. It has the advantage to provide a low-power highly accurate clock source to the real-time clock (RTC)

for clock/calendar or other timing functions. A crystal/resonator can be connected as shown in *Figure 38*: the crystal/resonator and the load capacitors must be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected crystal or ceramic resonator. Refer to the electrical characteristics section of the datasheet for more details.

The oscillator mode is selected by setting LSEBYP bit to 0 and LSEON bit to 1.

The LSE also offers a programmable driving capability (LSEDRV[1:0]) that can be used to modulate the amplifier driving capability. This driving capability is chosen according to the external crystal/ceramic component requirement to insure a stable oscillation.

The driving capability must be set before enabling the LSE oscillator.

Warning: It is not allowed to change the driving capability when the LSE is enabled. The LSE behavior is not guaranteed in that case.

LSE ready logic

The LSE offers an LSERDY flag which indicates whether the LSE clock is available or not. When the LSE is enabled (LSEON set to 1) the LSERDY flag goes to 1 when a certain amount of valid cycles of LSE clock have been detected. The **lse_ck** clock is not released until LSERDY goes to 1.

When LSEBYP = 0, the RCC waits 4096 clocks cycles before activating the LSERDY flag, when LSEBYP = 1, the RCC waits 8 clocks cycles.

An interrupt can be generated if enabled in the *RCC clock source interrupt enable register (RCC_CIER)*.

LSERDY bit is located into the *RCC Backup domain control register (RCC_BDCR)*.

LSE controls

The LSE control bits LSEBYP, LSEEXT, LSEDRV[1:0] and LSEON are located in the *RCC Backup domain control register (RCC_BDCR)*. This register is write-protected by DBP bit of PWR_CR1 register. In order to modify the RCC_BDCR register, the bit DBP must be set 1.

The LSE oscillator is switched ON and OFF using the LSEON bit.

The LSE remains enabled when the system enters Stop, Standby or VBAT mode (See *Table 53: Oscillator states versus system modes*).

The hardware does not allow changing the values of LSEBYP and LSEEXT when LSEON = 1.

In addition, the LSE clock can be driven to the MCO1 output and used as a clock source for external components.

LSE Programming sequence

In order to initialize the LSE, the application must follow the sequence hereafter:

- Set the DBP bit of PWR_CR1 to 1 in order to allow write accesses to the RCC_BDCR register.
- Disable the LSE by writing to 0 the LSEON bit,
- Check that the LSE is disabled by waiting until the bit LSERDY is set to 0,
- If the oscillator mode is needed:
  - Select the oscillator mode by setting LSEBYP to 0,
  - Configure LSEDRV[1:0],
- If an external clock is connected to OSC32_IN:
  - Select the bypass mode by setting LSEBYP to 1,
  - Set LSEEXT to 1
- Enable again the LSE by writing the LSEON bit to 1,
- Wait for LSERDY = 1, then the LSE is ready for use.
- If RCC_BDCR register no longer needs to be changed, set the DBP bit of PWR_CR1 to 0 in order to write-protect accesses.

Caution: If LSE bypass is used in V_{BAT} mode, the input clock signal must meet the required input levels relative to the V_{BAT} voltage. Refer to the device datasheet.

HSI oscillator

The HSI block provides the default clock to the product.

The HSI is a high-speed internal RC oscillator that can be used directly as system clock, peripheral clock, or as PLL input. A predivider allows the application to select an HSI output frequency of 8, 16, 32 or 64 MHz. This predivider is controlled by the HSIDIV.

The HSI advantages are the following:

- Low-cost clock source since no external crystal is required
- Faster startup time than HSE (a few microseconds)

The HSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

HSI controls

The HSI can be switched ON and OFF using the HSION bit.

The hardware does not allow modifying HSION bit if one of the following conditions is met:

- The HSI is used directly (via software mux) as the system clock.
- The HSI is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).
- The bit XSPICKP is set to 1.
- The bit FMCCKP is set to 1.

Note that the HSIDIV cannot be changed if the HSI is selected as reference clock for at least one enabled PLL (PLLxON bit set to 1). In that case the hardware does not update the HSIDIV with the new value. However it is possible to change the HSIDIV if the HSI is used directly as system clock.

The HSIRDY flag indicates if the HSI is stable or not. At startup, the HSI output clock is not released until this bit is set by the hardware.

The HSI clock can also be used as a backup source (auxiliary clock) if the HSE fails (refer to *Section : CSS on HSE*).

The HSI can be disabled or not when the system enters Stop mode (See *Table 53: Oscillator states versus system modes*).

In addition, the HSI clock can be driven to the MCO1 output and used as clock source for other application components.

Care must be taken when the HSI is used as kernel clock for communication peripherals, the application must take into account the following parameters:

- The time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available,
- The frequency accuracy.

HSION, HSIRDY and HSIDIV bits are located in the *RCC register map*.

HSI calibration

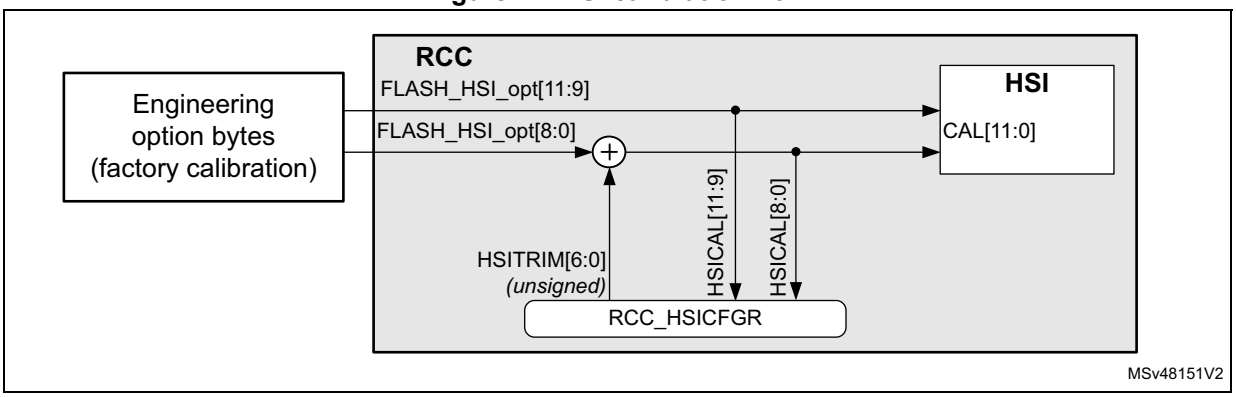
RC oscillator frequencies can vary from one chip to another due to manufacturing process variations. That is why each device is factory calibrated by STMicroelectronics to achieve an accuracy of ACC_{HSI} (refer to the product datasheet for more information).

After a power-on reset, or at the exit of Standby mode, the factory calibration value is loaded in the HSICAL[11:0] bits.

If the application is subject to voltage or temperature variations, this may affect the RC oscillator frequency. The user application can trim the HSI frequency using the HSITRIM[6:0] bits.

Note: HSICAL[11:0] and HSITRIM[6:0] bits are located in the RCC HSI calibration register (RCC_HSICFGR).

Figure 41. HSI calibration flow



CSI oscillator

The CSI is a low-power RC oscillator that can be used directly as system clock, peripheral clock, or PLL input.

The CSI advantages are the following:

- Low-cost clock source since no external crystal is required
- Faster startup time than HSE (a few microseconds)
- Very low-power consumption.

The CSI provides a clock frequency of about 4 MHz, while the HSI is able to provide a clock up to 64 MHz.

CSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

CSI controls

The CSI can be switched ON and OFF through the CSION bit. The CSIRDY flag indicates whether the CSI is stable or not. At startup, the CSI output clock is not released until this bit is set by the hardware.

The hardware does not allow modifying CSION bit if one of the conditions is met:

- The CSI is used directly (via software mux) as system clock.
- The CSI is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).
- The bit XSPICKP is set to 1.
- The bit FMCCKP is set to 1.

The CSI can be disabled or not when the system enters Stop mode (See *Table 53: Oscillator states versus system modes*).

In addition, the CSI clock can be driven to the MCO2 output and used as clock source for other application components.

Even if the CSI settling time is faster than the HSI, care must be taken when the CSI is used as kernel clock for communication peripherals: the application must take into account the following parameters:

- The time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available,
- The frequency precision.

Note: CSION and CSIRDY bits are located in the RCC register map.

CSI calibration

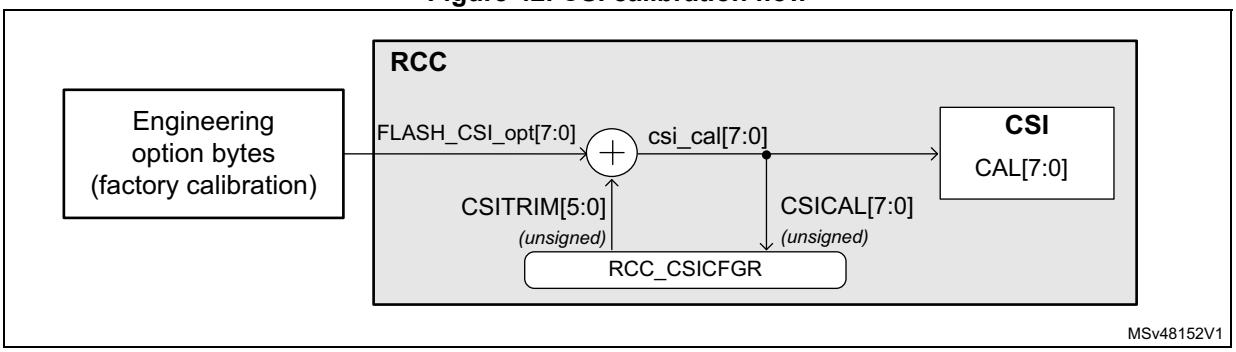
RC oscillator frequencies can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by STMicroelectronics to achieve accuracy of ACC_{CSI} (refer to the product datasheet for more information).

After a power-on reset, or at the exit of Standby mode, the factory calibration value is loaded in the CSICAL[7:0] bits.

If the application is subject to voltage or temperature variations, this may affect the RC oscillator frequency. The user application can trim the CSI frequency using the CSITRIM[5:0] bits.

Note: Bits CSICAL[7:0] and CSITRIM[5:0] are located into the RCC CSI calibration register (RCC_CSICFGR)

Figure 42. CSI calibration flow

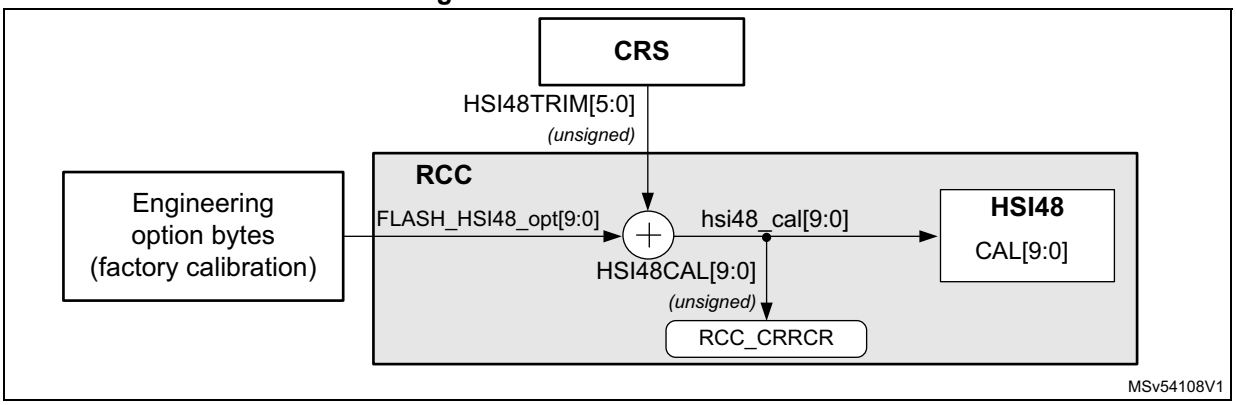


HSI48 oscillator

The HSI48 is an RC oscillator delivering a 48 MHz clock that can be used directly as kernel clock for some peripherals.

The HSI48 oscillator mainly aims at providing an accurate clock to the USB Full-Speed peripherals by means of a special clock recovery system (CRS) circuitry. The CRS can use the USB SOF signal, the LSE or an external signal to automatically adjust the oscillator frequency on-the-fly, with a very small granularity. It is possible to read the calibration value provided to the HSI48 via *RCC clock recovery RC register (RCC_CRRCR)*. The HSI48TRIM value can be adjusted through the CRS block, for more details, refer to *Section 8: Clock recovery system (CRS)*.

Figure 43. HSI48 calibration flow



The HSI48 oscillator is disabled as soon as the system enters Stop or Standby mode. When the CRS is not used, this oscillator is free-running and thus subject to manufacturing process variations. This is why each device is factory calibrated by STMicroelectronics to achieve an accuracy of ACC_{HSI48} (refer to the product datasheet for more information).

The HSI48RDY flag indicates whether the HSI48 oscillator is stable or not. At startup, the HSI48 output clock is not released until this bit is set by the hardware.

After a power-on reset, or at the exit of Standby mode, the factory calibration value is loaded into the HSI48.

The HSI48 can be switched ON and OFF using the HSI48ON bit.

The HSI48 clock can also be driven to the MCO1 multiplexer and used as clock source for other application components.

Note: HSI48ON and HSI48RDY bits are located in the RCC register map.

LSI oscillator

The LSI acts as a very-low-power clock source that can be kept running when the system is in Stop or Standby mode for the independent watchdog (IWDG) and auto-wakeup unit (AWU). The clock frequency is around 32 kHz. For more details, refer to the electrical characteristics section of the datasheet.

The LSI can be switched ON and OFF using the LSION bit. The LSIRDY flag indicates whether the LSI oscillator is stable or not. If an independent watchdog is started either by hardware or software, the LSI is forced ON and cannot be disabled.

The LSI remains enabled when the system enters Stop or Standby mode (See *Table 53: Oscillator states versus system modes*).

At LSI startup, the clock is not provided until the hardware sets the LSIRDY bit. An interrupt can be generated if enabled in the *RCC clock source interrupt enable register (RCC_CIER)*.

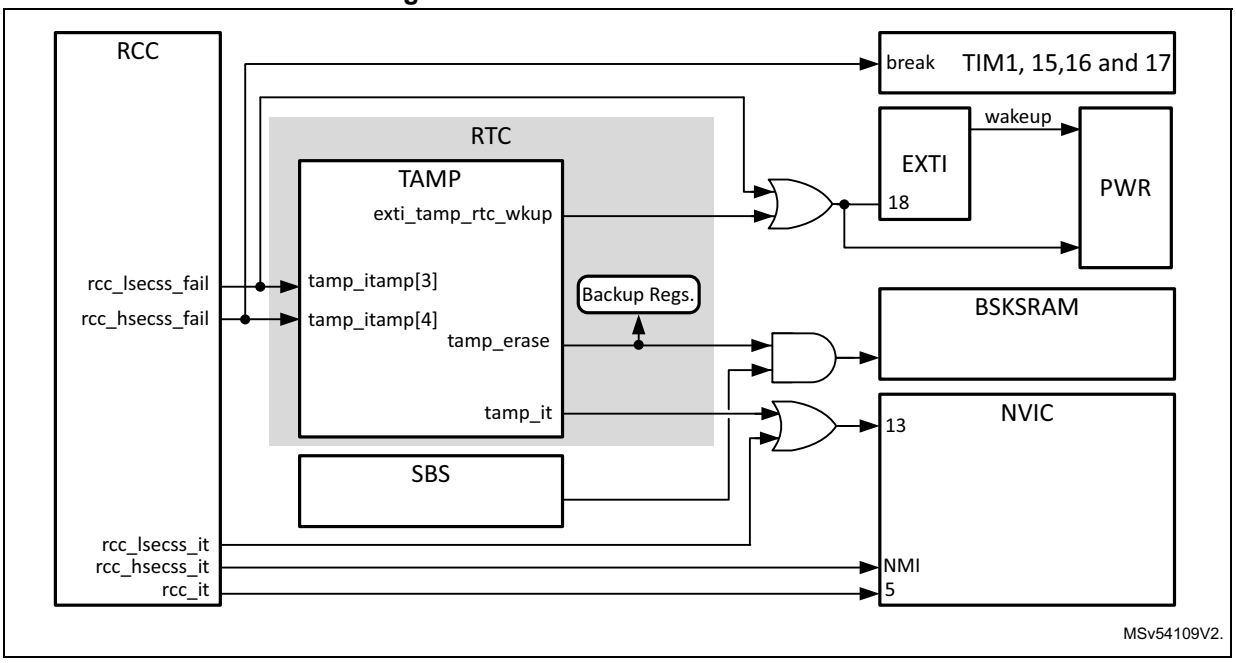
In addition, the LSI clock can be driven to the MCO2 output and used as a clock source for other application components.

Note: Bits LSION and LSIRDY bits are located into the RCC clock control and status register (RCC_CSR).

7.5.3 Clock security system (CSS)

The RCC offers a clock security system for the HSE and LSE oscillators. The clock security systems are capable of detecting a failure on HSE and LSE oscillators. The figure hereafter shows how this function interacts with other blocks.

Figure 44. LSE and HSE CSS function



CSS on HSE

The clock security system can be enabled by software via the HSECSSON bit. The HSECSSON bit can be enabled even when the HSEON is set to 0.

The CSS on HSE is activated when the HSE is enabled and ready, and the software sets the HSECSSON bit to 1. The CSS on HSE is no longer working when the HSE is disabled. For example, this function does not work when the system is in Stop mode.

It is not possible to clear directly the HSECSSON bit by software. The HSECSSON bit is cleared by hardware when a system reset occurs or when the system enters Standby mode (see *Section 7.4.2: The system and application resets (sys_rst and nreset)*).

When the CSS on HSE is enabled, the following actions are done by the RCC if a failure is detected:

- The signal **rcc_hsecss_fail** is asserted.
- If the STOPWUCK = 0, the HSI clock is enabled and the SW switch automatically selects HSI as system clock.
- If the STOPWUCK = 1, the CSI clock is enabled and the SW switch automatically selects CSI as system clock.
- The HSE is then automatically disabled.
- If the HSE output was used as clock source for PLLs, the PLLs are also disabled.
- The XSPISEL switch goes to recovery position.
- The FMCSEL switch goes to recovery position:
  - If PLL1 has been disabled and FMCSEL is in position 1.
  - If PLL2 has been disabled and FMCSEL is in position 2.

  See details on *Section : Clock distribution for SDMMC, FMC, and XSPI*.
- The clock failure event (**rcc_hsecss_fail**) is also sent to the break inputs of advanced-control timers (TIM1, TIM 8, TIM15, TIM16, TIM17 and TIM20) and high-resolution timers (HRTIM1, HRTIM2).
- An NMI interrupt is generated to inform the software about the failure (CSS interrupt: ***rcc_hsecss_it***), thus allowing the MCU to perform rescue operations. The CSS interrupt is asserted until the HSECSSF bit is cleared. The HSECSSF flag can be cleared by setting the HSECSSC bit in the *RCC clock source interrupt clear register (RCC_CICR)*.
- A tamper event can also be triggered, in order to clear content of the backup registers, and backup RAM.

CSS on LSE

A clock security system on the LSE oscillator can be enabled by software by programming the LSECSSON bit in the *RCC Backup domain control register (RCC_BDCR)*.

This bit is disabled by hardware if one of the following conditions is met:

- After a V_{SW} hardware reset (**pwr_vsw_rst),**
- After a V_{SW} software reset via VSWRST bit

The software can also disable the clock security system after an LSE failure detection.

The CSS on LSE works in all modes (Run, Stop and Standby) including VBAT.

The LSECSS provides a re-arm feature, offering to the software the possibility to re-arm the LSECSS and re-enable the LSE clock when a failure has been detected. This feature allows the application to decide if the LSE is to be provided again to the RTC even if a failure occurred, or if another action must be performed. For example the application can decide to reset the V_{SW} domain only if a certain number of consecutive LSE failures occurred, within a time window.

The LSECSS offers two flag signals:

- The LSECSSD, capable of retaining an LSE failures even in V_{BAT} mode.
- The LSECSSF is used to generate interrupt in case of LSE failure. This flag is not affected by a failure detected when the product is in V_{BAT} mode

The sequence hereafter describes the LSE enabling sequence with the clock security system enabled.

- Follow the LSE enable procedure given is the section *LSE Programming sequence*, except the last step
- Select the LSE clock via RTCSEL[1:0]
- Set the LSECSSON bit to 1
- If the RCC_BDCR register no longer needs to be changed, set the DBP bit of PWR_CR1 to 0 in order to write-protect accesses.

Note: The LSECSSON bit must be enabled after the LSE is enabled (LSEON bit set by software) and ready (LSERDY set by hardware), and after the RTC clock has been selected through the RTCSEL bit.

If a failure is detected on the LSE, the hardware does the following:

- The LSE clock is no more delivered to the RTC.
- The values of RTCSEL, LSECSSON and LSEON bits are not changed by the hardware.
- A failure event is generated (**rcc_lsecss_fail**). This event allows the system to wake up from Standby, but also to protect the backup registers and BKPSRAM via TAMP block. This event is also generated in VBAT mode.
- The interrupt flag LSECSSF is activated (except in V_{BAT} mode) in order to generate an interrupt (**rcc_lsecss_it**).
- The LSECSSD flag is activated as well, retaining the first LSE failure even in V_{BAT} mode.

In the software side, different actions can be taken according to the application requirements. Three different cases are described hereafter in order to illustrate the hardware behavior, they can also be combined. The application can also decide to handle LSE failure differently.

Case A:

The application no longer wants to use LSE when a failure is detected:

- Unlock the RCC_BDCR register by setting the DBP bit of PWR_CR1 to 1
- Disable the CSS function (this step is mandatory)
  - Clear the LSECSSF bit, if interrupt was enabled for this event
  - Set the LSECSSON bit to 0
  - Set the LSEON bit to 0, in order to disable the LSE
- Change the clock source for the RTC if needed:
  - Set the RTCEN bit to 0 to disable the RTC clock
  - Enable the new clock source for the RTC
  - Select the proper clock source via RTCSEL
  - Set the RTCEN bit to 1 to enable the RTC clock
- The application must perform specific actions consecutive to TAMPER events if enabled
- Lock the RCC_BDCR register by setting the DBP bit of PWR_CR1 to 0

Case B:

The application wants to re-initialize the V_{SW} domain:

- Unlock the RCC_BDCR register by setting the DBP bit of PWR_CR1 to 1
- Perform a V_{SW} reset by setting the VSWRST bit to 1, then back to 0 (it is in reality a pulse of one period of sys_ck)
- The application must perform specific actions consecutive to TAMPER events if enabled
- Re-initialize all components of the V_{SW} domain.
- Lock the RCC_BDCR register by setting the DBP bit of PWR_CR1 to 0

Case C:

The application tries to re-use LSE when a failure is detected:

- If the number of failures in a given time window is higher than a given threshold then go to case A or B. Otherwise continue to the next step.
- Unlock the RCC_BDCR register by setting the DBP bit of PWR_CR1 to 1
- Clear the LSECSSF bit if interrupt was enabled for this event.
- The application must perform specific actions consecutive to TAMPER events if enabled
- Set the LSECSSON bit to 0.
- Re-arm the LSECSS function by writing the LSECSSRA bit to 1, then back to 0.
- Wait for LSERDY set 1. The LSERDY flag must go to 1 after the oscillator settling time delay plus, 4096 periods of LSE clock. If this is not the case, it probably means that the LSE failure is permanent. It is not possible to set LSECSSON to 1, it is recommended to execute case A or B.
- Set the LSECSSON bit to 1.
- When LSECSSON is seen to 1, the LSE is enabled, and protected by LSECSS.
- Lock the RCC_BDCR register by setting the DBP bit of PWR_CR1 to 0

All bits used in these sequences (except DBP) are located into *RCC Backup domain control register (RCC_BDCR)*.

7.5.4 Clock output generation (MCO1/MCO2)

Two microcontroller clock output pins MCO1 and MCO2, are available. A clock source can be selected for each output. The selected clock can be divided thanks to a configurable prescaler (refer to *Figure 37* for additional information on signal selection).

MCO1 and MCO2 outputs are controlled via MCO1PRE[3:0], MCO1SEL[2:0], MCO2PRE[3:0] and MCO2SEL[2:0] located in the *RCC clock configuration register (RCC_CFGR)*.

The GPIO port corresponding to each MCO pin must be programmed in alternate function mode.

The dividers MCO1PRE and MCO2PRE, provide a clock with a duty cycle of 50% for even divisions values. More generally the duty cycle is given by the following formula:

$$DC(\%) = \frac{\text{FLOOR}\left(\frac{\text{MCOxPRE}}{2}\right)}{\text{MCOxPRE}} \times 100$$

For MCOxPRE = 2 to 15.

Note that the MCO1 and MCO2 outputs are available in Run and Stop modes.

Caution: The clock provided to the MCOs outputs must not exceed the maximum PAD speed, refer to the product datasheet for information about the supported PAD speed.

The table hereafter shows the signals available on each MCO output.

Table 54. Oscillator states versus system modes

MCO1SEL		MCO2SEL	
Pos.	**Clock source**	**Pos.**	**Clock source**
0	hsi_ck	0	sys_ck
1	lse_ck	1	pll2_p_ck
2	hse_ck	2	hse_ck
3	pll1_q_ck	3	pll1_p_ck
4	hsi48_ck	4	csi_ck
-	-	5	lsi_ck

7.5.5 PLL description

The RCC features three PLLs:

- PLL1, generally used to provide clocks to the CPU, busses, and some peripherals
- PLL2, generally used to provide clocks to XSPI and SDMMC, and some peripherals
- PLL3, dedicated to the kernel clock generation for peripherals

The PLLs integrated into the RCC are completely independent. They offer the following features:

- A VCO supporting two modes:
  - A wide-range
  - A low-range used for instance in audio application cases
- Input frequency range:
  - 2 to 16 MHz for the VCO in wide-range mode
  - 1 to 2 MHz for the VCO in low-range mode
- VCO frequency range:
  - 384 to 1672 MHz for the VCO in high-range mode (VCOH), before the divider.
  - 150 to 420 MHz for the VCO in low-range mode (VCOL)
- Capability to work either in integer with or without SSGC, or in fractional mode
- 13-bit sigma-delta modulator, allowing to fine-tune the VCO frequency by steps of 10 to 0.3 ppm
- The sigma-delta modulator can be updated on-the-fly without generating frequency overshoots on PLLs outputs.
- PLLs offer up to 5 outputs with post-dividers.

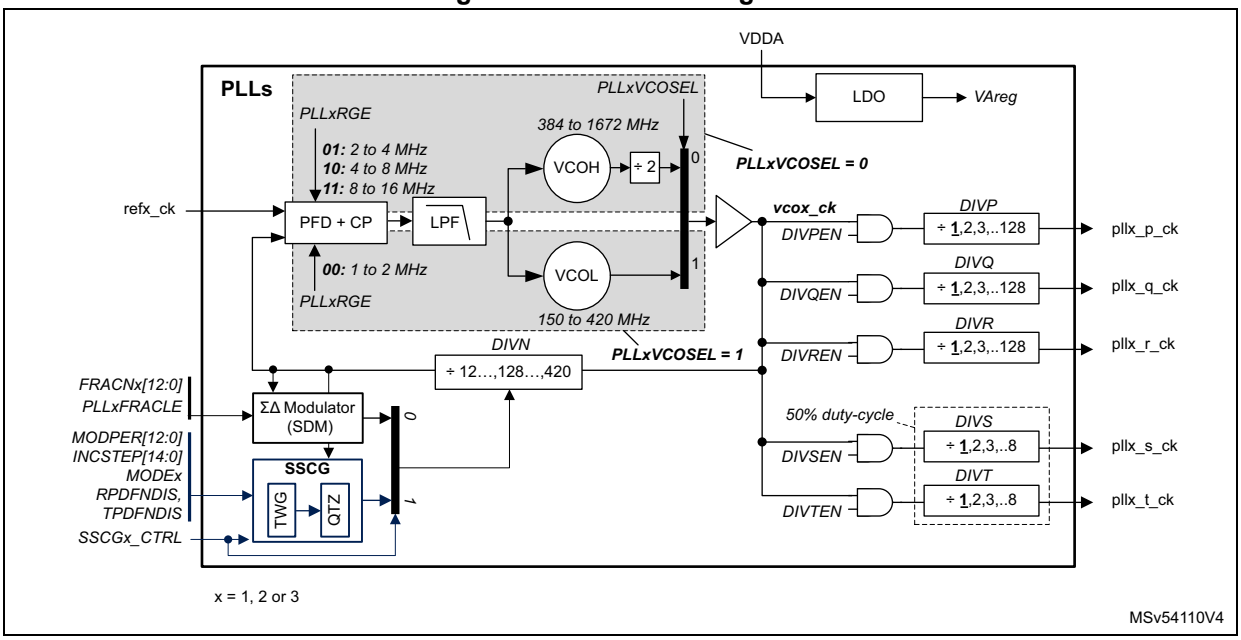
For better flexibility, all PLLs offer two VCOs (VCOH and VCOL), the application can select the wanted VCO via PLLxVCOSEL bit.

The frequency of the reference clock provided to the PLLs (**refx_ck**) must range from 1 to 16 MHz. The DIVMx dividers of the *RCC PLLs clock source selection register (RCC_PLLCKSELR)* must be properly programmed in order to match this condition. In addition, the PLLxRGE[1:0] field of the *RCC PLLs configuration register (RCC_PLLCFGR)* must be set according to the reference input frequency to guarantee an optimal performance of the PLL.

The user application can then configure the proper VCO. VCOL must be chosen when the reference clock frequency is lower than 2 MHz.

To reduce the power consumption, it is recommended to configure the VCO output to the smaller range.

Figure 45. PLL block diagram



The PLLs can be enabled by setting PLLxON to 1. The PLLxRDY bits indicate that the PLL is ready (means locked).

The DIVN loop divider must be programmed to achieve the expected frequency at VCO output. The VCO output range must be respected.

PLLs have a divider-by-2, at VCOH output, insuring that the clock provided to the post-dividers has a duty-cycle of 50%.

The frequency of the clocks provided by the PLLs can also be adjusted thanks to post-divider DIVP, DIVQ, DIVR, DIVS and DIVT. The post-divider values can be changed without disabling the PLLs, if their respective enable bits are set to 0.

The post-dividers provide clocks with a 50% duty-cycle in the following conditions:

- When the post-dividers are dividing the clock by an even value.
- When VCOH is used and post-dividers are bypassed.
- When VCOH is selected and post-dividers DIVS and DIVT are used.

PLL programming recommendations

This section is providing a list of recommendations to follow in order to guarantee a good usage of the PLLs. Note that programming examples are given in *Section 7.7.1: PLL programming procedure*.

- Before enabling the PLLs, the user must ensure that the reference frequency (**refx_ck**) provided to the PLL is stable and in the good range.
- When one or several PLL outputs are not used, the application must set the corresponding enable bit DIV[y]ENx and the divider value DIV[y]x to 0 (with x = 1, 2 or 3, and y = P, Q, R, S or T).
- In order to ensure the good behavior of the PLL, the bits PLLxPEN, PLLxQEN, PLLxREN, PLLxSEN and PLLxTEN must be set to 0 before disabling the corresponding PLL. In the same way, the bits PLLxPEN, PLLxQEN, PLLxREN,

PLLxSEN and PLLxTEN can be set to 1 only if the corresponding bit PLLON is already set to 1 and the PLL is ready.

Refer to section *Section 7.7.1: PLL programming procedure* for additional information.

PLL protections

- The RCC hardware does not allow changing DIVMx, DIVN, PLLxRGE, PLLxVCOSEL when the corresponding PLLx is ON and it is also not possible to change PLLSRC when one of the PLL is ON.
- The hardware prevents writing PLL1ON or PLL1PEN to 0 if the PLL1 is currently used to deliver the system clock. There are other hardware protections on the clock generators (refer to *Section 7.5.7: Clock protection*).

The PLLs are disabled by hardware when the system enters Stop or Standby mode.

PLLs using HSE as reference clock are also disabled by hardware if an HSE failure is detected.

Note: The refx_ck clock is provided to the PLLx when the corresponding PLLON bit is set to 1.

The PLLs can work in 3 different modes:

- In integer mode
- In fractional mode
- In spread spectrum mode

Using the PLLs in integer mode

The PLL is working in integer mode when the sigma-delta modulator (SDM) is loaded with a 0, and the bit PLLxSSCGEN = 0.

In order to load 0 into the SDM, the user has to perform the following sequence:

– Write PLLxFRACLE of *RCC PLLs configuration register (RCC_PLLCFGR)* to 0
– Write FRACN[12:0] to 0,
– Write PLLxFRACLE to 1
– Wait at least 5 µs

The bits FRACN[12:0] are located in the RCC_PLLxFRACR registers.

The VCO frequency (F_{VCO}) and output frequencies expressions are given by the following table:

Table 55. VCO frequency and output frequency in integer mode

PLLxVCOSEL = 0	PLLxVCOSEL = 1
$F_{VCO} = 2 \times F_{ref_ck} \times (DIVN + 1)$	$F_{VCO} = F_{ref_ck} \times (DIVN + 1)$
$F_{pll_y_ck} = \frac{F_{VCO}}{2 \times (DIVy + 1)}$ with y = P, Q, R, S or T	$F_{pll_y_ck} = \frac{F_{VCO}}{(DIVy + 1)}$ with y = P, Q, R, S or T

Refer to section *Section 7.7.1: PLL programming procedure* for additional information.

Using the PLLs in fractional mode

The fractional mode is activated when the value loaded into the SDM is different from 0, and the bit PLLxSSCGEN = 0.

The SDM value can be updated at any time by the application by executing the following sequence:

- Write PLLxFRACLE bit to 0
- Wait at least three reference clock cycles
- Write into FRACN[12:0] the new fractional value,
- Write PLLxFRACLE to 1.
  The new FRACN value is applied after maximum three reference clock periods

The sigma delta modulator is designed in order to minimize the jitter impact while allowing very small frequency steps.

When the PLL is used in fractional mode, the DIVN divider must be initialized before enabling the PLL.

This feature can be used either to generate a specific frequency, with a good accuracy from any crystal value, or to perform a fine-tuning of the frequency on-the-fly.

The VCO frequency (F_{VCO}) and output frequencies expressions are given by the following table:

Table 56. VCO frequency and output frequency in fractional mode

PLLxVCOSEL = 0	PLLxVCOSEL = 1
$F_{VCO} = 2 \times F_{ref_ck} \times \left(DIVN + 1 + \frac{FRACN}{2^{13}} \right)$	$F_{VCO} = F_{ref_ck} \times \left(DIVN + 1 + \frac{FRACN}{2^{13}} \right)$
$F_{pll_y_ck} = \frac{F_{VCO}}{2 \times (DIVy + 1)}$ with y = P, Q, R, S or T	$F_{pll_y_ck} = \frac{F_{VCO}}{(DIVy + 1)}$ with y = P, Q, R, S or T

Refer to section *Section 7.7.1: PLL programming procedure* for additional information.

Using the PLLs in spread spectrum mode

The spread spectrum mode is activated when the SDM is loaded with 0, and the bit PLLxSSCGEN = 1. This feature is available on all PLLs.

The VCOH must be selected when the spread spectrum mode is activated. VCOH is selected by setting PLLxVCOSEL to 0. PLLxVCOSEL are located into the *RCC clock configuration register (RCC_CFGR)*.

The spread spectrum technique consists of modulating the VCO frequency with a low-frequency signal (in our case a triangular signal), in order to spread the clock energy into a wider frequency band. As a consequence, the number of emitted EMI peaks is reduced.

The parameters of the spread spectrum modulation are adjusted via the following fields:

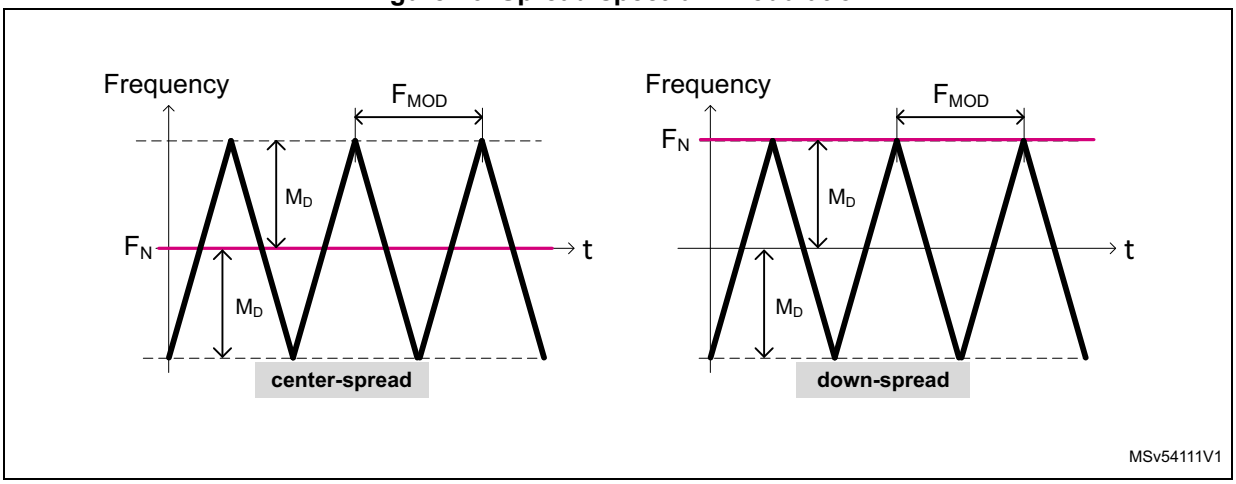
- MODPER[12:0]: used to adjust the modulation frequency
- INCSTEP[14:0]: used to adjust the modulation depth (or modulation index)
- SPREADSEL: used to define if the modulation is centered around the VCO frequency (center-spread) or if it is lowered with respect to VCO frequency (down-spread).

The bits MODPER[12:0], INCSTEP[14:0] and SPREADSEL are located into the RCC_PLLxSSCGR registers with x=1, 2 or 3.

Figure 46 shows the signal modulating the nominal frequency (F_N), when SPREADSEL = 0 (center-spread) and SPREADSEL = 1 (down-spread). The nominal frequency is the frequency output by the PLL in integer mode, when no clock spreading is applied.

Setting the SPREADSEL bit to 1 (down-spread) guarantees that the PLL output frequency does not exceed the programmed frequency value when the SSCG is enabled.

Figure 46. Spread-spectrum modulation



The peak modulation depth (in percentage) is given by the following formula:

$$M_D\,(\%) = \frac{\text{MODPER} \times \text{INCSTEP} \times 100 \times 5}{(2^{15} - 1) \times (\text{DIVN} + 1)}$$

Note that MODPER x INCSTEP must not exceed (2^{15}-1).

The modulation frequency (F_{mod}) is given by:

$$F_{mod} = \frac{F_{ck_ref}}{4 \times \text{MODPER}}$$

Note: Refer to the datasheet of the product for the recommended modulation frequency range.

In order to use the spread spectrum feature, the user has to do the following:

- Program the PLL as explained in *Section : Using the PLLs in integer mode* in order to adjust the nominal frequency (F_N) according to the targeted by the application.
- Compute the MODPER value according to the wanted modulation frequency (F_{mod}):

$$MODPER = ROUND\left(\frac{F_{ck_ref}}{4 \times F_{mod}}\right)$$

- Compute the INCSTEP value according to the wanted modulation depth (M_D):

$$INCSTEP = ROUND\left(\frac{(2^{15} - 1) \times M_D \times (DIVN + 1)}{100 \times 5 \times MODPER}\right)$$

  Check that MODPER x INCSTEP does not exceed (2^{15}-1).
- Then DIVP, Q, R, S or T can be adjusted, and the bit PLLxSSCGEN can be set to 1, and the PLL enabled.

The user can check F_{MIN}, F_{MAX} and F_C as follows:

- If SSCG_MODE = 0 (centered-spread)

  F_C is given by the formula of the *Section : Using the PLLs in integer mode*

  $F_{MIN} = F_C$ x (1 - M_D/100)

  $F_{MAX} = F_C$ x (1 + M_D/100)
- If SSCG_MODE = 1 (down-spread)

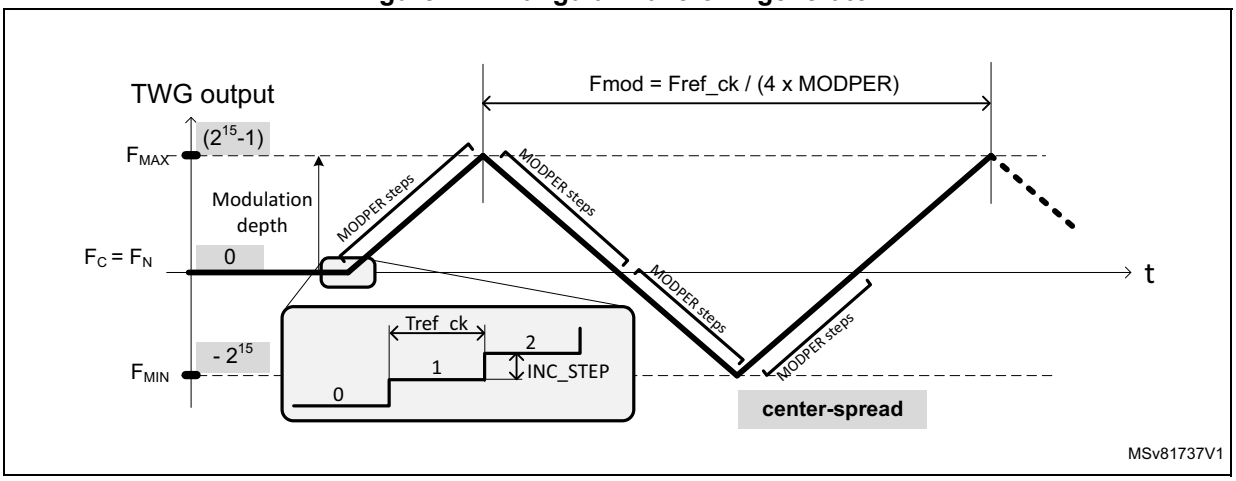
  F_{MAX} is given by the formula of the *Section : Using the PLLs in integer mode*

  $F_{MIN} = F_{MAX}$ x (1 - 2 x M_D/100)

  $F_C = F_{MAX}$ x (1 - M_D/100)

Figure 47 shows the digital signal generated by Triangular Waveform Generator (TWG block), and the way of MODPER and INCSTEP are changing the triangular waveform.

Figure 47. Triangular waveform generator



Refer to section *Section 7.7.1: PLL programming procedure* for additional information.

7.5.6 System clock (sys_ck)

System clock selection

After a system reset, the HSI is selected as system clock and all PLLs are switched OFF. When a clock source is used for the system clock, it is not possible for the software to disable the selected source via the xxxON bits.

Of course, the system clock can be stopped by the hardware when the system enters Stop or Standby mode.

When the system is running, the user application can select the system clock (**sys_ck**) among the 4 following sources:

- HSE
- HSI
- CSI
- pll1_p_ck

This function is controlled by programming the *RCC clock configuration register (RCC_CFGR)*. A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source that is not yet ready is selected, the switch occurs when the clock source is ready.

The SWS status bits in the *RCC clock configuration register (RCC_CFGR)* indicate which clock is currently used as system clock. The other status bits in the RCC_CR register indicate which clock(s) is (are) ready.

System clock generation

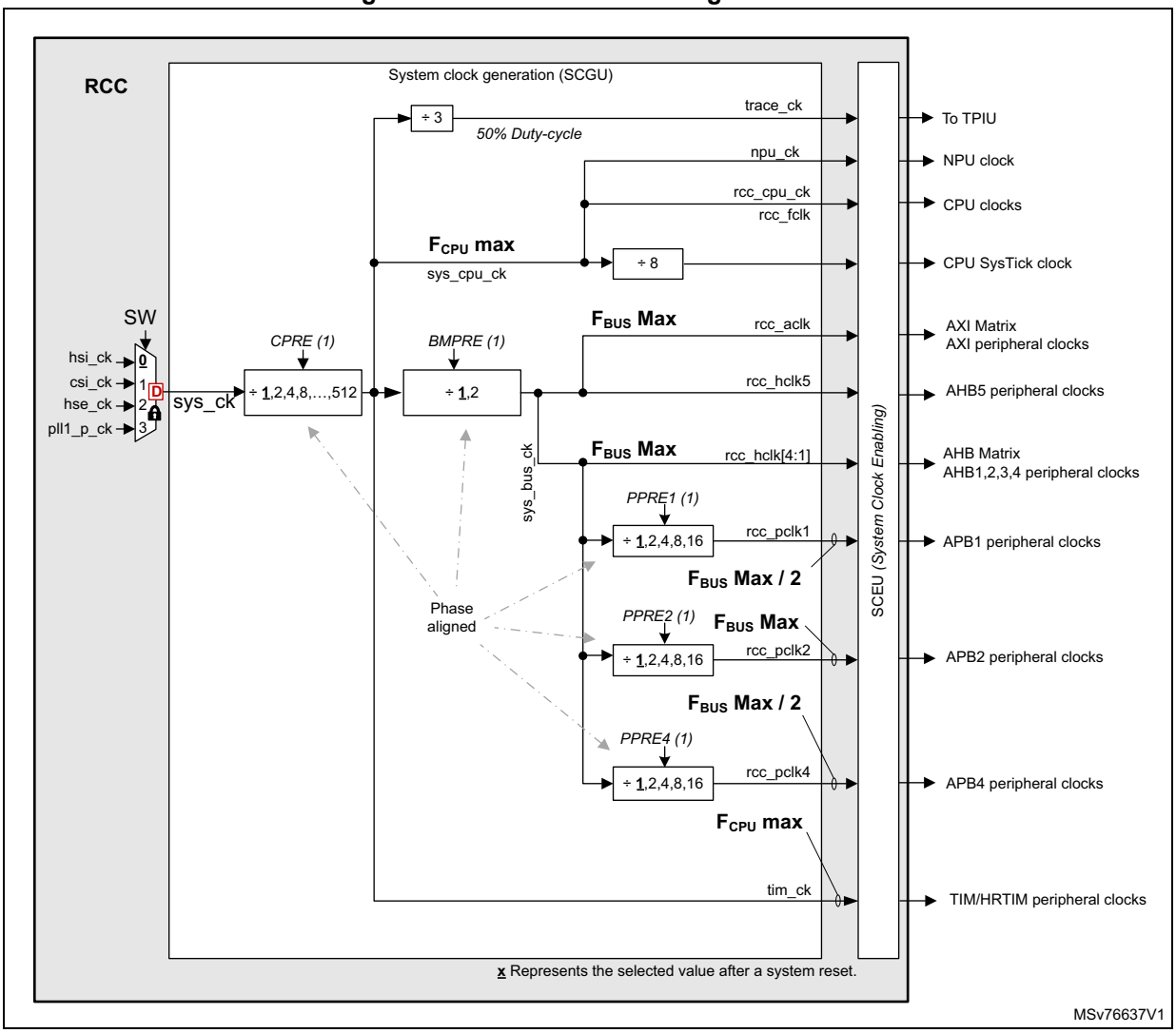
Figure 48 shows a simplified view of the clock distribution for the CPU and buses. All the dividers shown in the block diagram can be changed on-the-fly without generating timing violations. This feature is a very simple solution to adapt the bus frequencies to the application needs, thus optimizing the power consumption.

The CPRE divider can be used to adjust the CPU clock. However this also impacts the clock frequency of all bus matrix. CPRE is controlled via the RCC_CDCFGR register.

In the same way, BMPRE divider can be used to adjust the clock for the bus matrix (AHB and AXI), but this also impacts the clock frequency of APB buses. BMPRE is controlled via the RCC_BMCFGR register.

Most of the prescalers are controlled via RCC_CDCFGR and RCC_SRDCFGR registers.

Figure 48. Core and bus clock generation



1. Divider values can be changed on-the-fly. All dividers have 50% duty cycles.

Note that the application must respect the maximum allowed frequencies: F_{CPU} Max and F_{BUS} Max. F_{BUS} represents the maximum allowed frequency for the AHB and AXI buses. Refer to the product datasheet for maximum values.

*Note: The trace clock (**trace_ck**) is generated from **sys_cpu_ck** clock, divided by 3. For additional information refer to Section : Clock distribution for Debug and Trace.*

7.5.7 Clock protection

The RCC does not allow the system clock (**sys_ck**) to be disabled. Several protective measures are implemented to prevent accidentally disabling the system clock.

In addition, protection bits are provided to prevent accidental disabling of FMC and XSPI block activities.

The protection consists in ignoring invalid write operations. It is recommended to read back registers having protection to insure that the clock configuration expected by the software is accepted by the RCC.

The table hereafter shows the protections handled by the RCC.

Table 57. Clock protection summary

Protected fields	Conditions checked by the RCC
SW[2:0]	Changing SW value is allowed only if the targeted clock is enabled and ready.
PLLSRC[1:0]	Changing PLLSRC value is allowed if: - The targetted clock is enabled and ready, and, - All PLL disabled
HSEON, HSION or CSION	Changing these fields is allowed if: - The oscillator is not used directly or indirectly (via PLL1) as system clock, and - FMCCKP = XSPICKP = 0.
DIVMx[5:0], PLLxRGE[1:0], PLLxVCOSEL, DIVN[8:0]	Changing these fields is allowed only if the corresponding PLL is disabled (PLLxON=0)
DIVP[6:0], DIVQ[6:0], DIVR[6:0], DIVS[2:0], DIVT[2:0]	Changing these fields is allowed only if the corresponding DIVyEN bit is equal to 0 (y = P, Q, R, S or T)
PLL1ON	Changing this field is allowed if: - SW switch is not selecting the PLL1, and - FMCCKP = 0.
PLL1PEN	Disabling the DIVP output of PLL1 is allowed if the SW switch is not selecting PLL1
PLL2ON	Changing this field is allowed only if FMCCKP = XSPICKP = 0.
PLL2SEN, PLL2TEN, XSPI1EN, XSPI1LPEN XSPI1RST	Changing these fields is allowed only if XSPICKP = 0.
PLL1QEN, PLL2REN, FMCEN, FMCLPEN, FMCRST	Changing these fields is allowed only if FMCCKP = 0.
HSEBYP, HSEEXT	Changing these fields is allowed only if HSEON = 0.
LSEBYP, LSEEXT	Changing these fields is allowed only if LSEON = 0.

There is also a clock protection mechanism in case of HSE failure, refer to *Section : CSS on HSE*.

7.5.8 Clock generation in Stop and Standby modes

When the whole system enters Stop mode, all the clocks (system and kernel clocks) are stopped and the following clock sources are disabled as well:

- CSI, HSI (depending on HSIKERON, and CSIKERON bits)
- HSE
- PLL1, PLL2 and PLL3
- HSI48

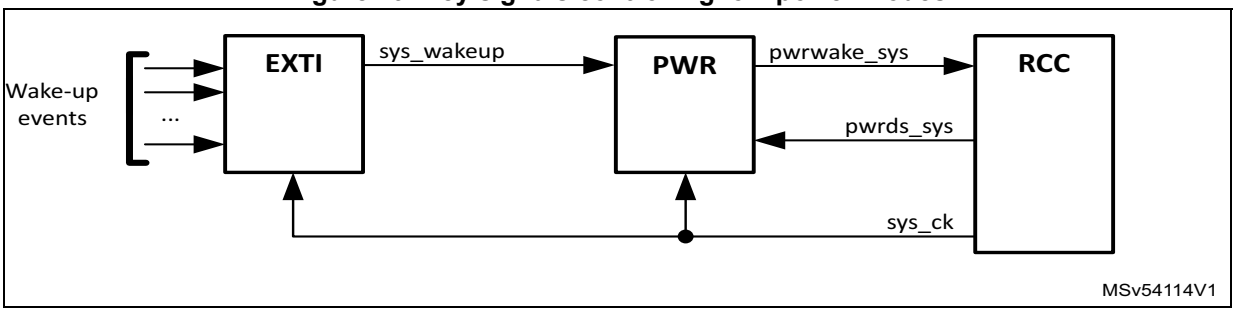
The content of the RCC registers is not altered except for PLL1ON, PLL2ON, PLL3ON HSEON and HSI48ON that are set to 0.

HSION and CSION bits are also affected depending on STOPWUCK and STOPKERWUCK bits (see *Table 58*).

When the CPU requests to go into Stop mode, the RCC first stops all the requested clocks, and informs the PWR that all clocks have been properly stopped. As shown in *Figure 49*, three main signals are used to control power transitions:

- The **rcc_pwrds** is used to indicate to the PWR that the RCC has stopped all the clocks, and thus the PWR can go to Stop or Standby.
- The **pwr_wkup** is used to indicate to the RCC to re-enable the clocks.
- The **exti_wkup** is used to indicate to the PWR that an event requests the system to exit from Stop mode.

Figure 49. Key signals controlling low-power modes



Exiting Stop mode

When the microcontroller exits system Stop mode via a wakeup event, the application can select which oscillator (HSI and/or CSI) is used to restart. The STOPWUCK bit selects the oscillator used as system clock. The STOPKERWUCK bit selects the oscillator used as kernel clock for peripherals. The STOPKERWUCK bit is useful if after a system Stop, a peripheral needs a kernel clock generated by an oscillator different from the one used for the system clock.

All these bits belong to the *RCC clock configuration register (RCC_CFGR)*.

Table 58 gives a detailed description of their behavior.

Table 58. STOPWUCK and STOPKERWUCK description

STOPWUCK	STOPKERWUCK	-	Activated oscillator when system exits Stop mode	Distributed clocks when system exits Stop mode	
				System clock	Kernel clock
0	0	→	HSI	HSI	HSI
	1	→	HSI and CSI		HSI and/or CSI
1	0	→		CSI	
	1	→	CSI		CSI

During Stop mode

There are two specific cases where the HSI or CSI can be enabled during system Stop mode:

- When a dedicated peripheral requests the kernel clock
  The peripheral receives the HSI or CSI according to the kernel clock source selected for this peripheral (via PERxSRC).
- When the HSIKERON or CSIKERON bits are set the HSI and CSI are kept running during Stop mode but the outputs are gated. The clock is then available immediately when the system exits Stop mode or when a peripheral requests the kernel clock (see *Table 59* for details).

HSIKERON and CSIKERON bits belong to *RCC register map*. *Table 59* gives a detailed description of their behavior.

Table 59. HSIKERON and CSIKERON behavior

HSIKERON (CSIKERON)	-	HSI (CSI) state during Stop mode	HSI (CSI) setting time
0	→	OFF	$t_{su(HSI)}$ ($t_{su(CSI)}$) (1)
1	→	Running and gated	Immediate

1. $t_{su(HSI)}$ and $t_{su(CSI)}$ are the startup times of the HSI and CSI oscillators (refer to the product datasheet for values of these parameters).

When the microcontroller exits Standby mode, the HSI is selected as system and kernel clock. The RCC registers are reset to their initial values except for the RCC_RSR and RCC_BDCR registers.

Note that the HSI and CSI outputs provide two clock paths (see *Figure 37*):

- One path for the system clock (**hsi_ck** or **csi_ck**)
- One path for the peripheral kernel clock (**hsi_ker_ck** or **csi_ker_ck**).

When a peripheral requests the kernel clock in system Stop mode, only the path providing the **hsi_ker_ck** or **csi_ker_ck** is activated.

Caution: The CPU does not automatically get the same clock frequencies when leaving Stop mode: it is up to the application to restore the previous clock settings if needed.

7.5.9 Peripheral clock distribution

Some peripherals are designed to work with two different clock domains that operate asynchronously:

- A clock domain synchronous with the register and bus interface (**ckg_bus_perx** clock)
- A clock domain generally synchronous with the peripheral (kernel clock)

The benefit of having peripherals supporting these two clock domains is that the user application has more freedom to choose an optimized clock frequency for the CPU, bus matrix and for the kernel part of the peripheral.

As a consequence, the user application can change the bus frequency without reprogramming the peripherals. As an example an ongoing transfer with UART is not disturbed if its APB clock is changed on-the-fly.

Table 60 shows the clocks the RCC delivers to the peripherals. Note as well that the clock named per_ck, is the output of a mux which allows the application to select:

- hsi_ker_ck, or
- csi_ker_ck, or
- hse_ker_ck

Table 60. Peripheral clock distribution summary

Peripherals	Clock types	Clock sources	Kernel clock MUX		Max Kernel clock freq. [MHz] (1)	Type (2)(3)
			Pos.	Control field		
ADF	Kernel	hclk1	**0**(4)	ADF1SEL	F_{MAX} / 2	A
		pll2_p_ck	1			
		pll3_p_ck	2			
		AUDIOCLK	3			
		csi_ker_ck	4			
		hsi_ker_ck	5			
	Bus	hclk1	-	-	F_{MAX} / 2	-
ADC12,34	Kernel	hclk3	**0**(4)	ADCSEL	75	A
		rcc_tim_ker_ck	1			
		hsi_ker_ck	2			
		hse_ker_ck	3			
		pll2_p_ck	4			
		pll3_r_ck	5			
		per_ck	6			
	Bus	hclk3	-	-	F_{MAX} / 2	-
COMP12,34,56,78	Bus	pclk4	-	-	F_{MAX} / 4	-
CORDIC	Bus	hclk2	-	-	F_{MAX} / 2	-
CRC	Bus	hclk4	-	-	F_{MAX} / 2	-
CRS	Bus	pclk1	-	-	F_{MAX} / 4	-

Table 60. Peripheral clock distribution summary

Peripherals	Clock types	Clock sources	Kernel clock MUX		Max Kernel clock freq. [MHz] (1)	Type (2)(3)
			Pos.	Control field		
DAC1,2,3,4	Kernel	rcc_hclk3	**0**(4)	DACxSEL	F_{MAX}	A
		rcc_tim_ker_ck	1			
		rcc_adc_ker_ck	2			
		hsi_ker_ck	3			
		hse_ker_ck	4			
		pll2_p_ck	5			
		pll3_r_ck	6			
		per_ck	7			
	Bus	hclk3			F_{MAX} / 2	
DBG	Bus	sys_bus_ck	-	-	F_{MAX} / 2	-
	kernel	sys_cpu_ck/3	-	-	F_{MAX} / 3	-
DB_XSPI1	Bus	hclk5	-	-	F_{MAX} / 2	-
DB_SDMMC1	Bus	hclk5	-	-	F_{MAX} / 2	-
FMAC1	Bus	hclk2	-	-	F_{MAX} / 2	
GPDMA1	Bus	hclk1	-	-	F_{MAX} / 2	-
HPDMA1	Bus	hclk5	-	-	F_{MAX} / 2	-
		aclk				
EXTI	Bus	pclk4	-	-	F_{MAX} / 4	-
FLASH	Bus	hclk5	-	-	F_{MAX} / 2	-
FDCAN	Kernel	hse_ker_ck	**0**(4)	FDCANSEL	125	A
		pll1_q_ck	1			
		pll2_p_ck	2			
	Bus	pclk1	-	-	F_{MAX} / 4	-
FMC	Kernel	hclk5	**0**(4)	FMCSEL	F_{MAX} / 2	A
		pll1_q_ck	1			
		pll2_r_ck	2			
		hsi_ker_ck	3			
	Bus	hclk5	-	-	F_{MAX} / 2	-
		aclk				
GPIOA-I	Bus	hclk4	-	-	F_{MAX} / 2	-
HASH	Bus	hclk3	-	-	F_{MAX} / 2	-
HRTIM1, HRTIM2	Kernel	tim_ck	-	-	F_{MAX}	S
	Bus	pclk2	-	-	F_{MAX} / 2	-

Table 60. Peripheral clock distribution summary

Peripherals	Clock types	Clock sources	Kernel clock MUX		Max Kernel clock freq. [MHz] (1)	Type (2)(3)
			Pos.	Control field		
HSP	Bus	hclk1	-	-	F_{MAX} / 2	-
I2C2, I2C3	Kernel	pclk1	**0**(4)	I2C23SEL	F_{MAX} / 4	A
		pll3_r_ck	1			
		hsi_ker_ck	2			
		csi_ker_ck	3			
	Bus	pclk1	-	-	F_{MAX} / 4	-
I2C1, I3C1	Kernel	pclk1	**0**(4)	I2C1_I3C1SEL	F_{MAX} / 4	A
		pll3_r_ck	1			
		hsi_ker_ck	2			
		csi_ker_ck	3			
	Bus	pclk1	-	-	F_{MAX} / 4	-
IWDG	Kernel	lsi_ck	-	-	1	A
	Bus	pclk4	-	-	F_{MAX} / 4	-
LPTIM1	Kernel	pclk1	**0**(4)	LPTIM1SEL	F_{MAX} / 4	A
		pll2_p_ck	1			
		pll3_r_ck	2			
		lse_ck	3			
		lsi_ck	4			
		per_ck	5			
	Bus	pclk1	-	-	F_{MAX} / 4	-
LPTIM2, LPTIM3	Kernel	pclk4	**0**(4)	LPTIM23SEL	F_{MAX} / 4	A
		pll2_p_ck	1			
		pll3_r_ck	2			
		lse_ck	3			
		lsi_ck	4			
		per_ck	5			
	Bus	pclk4	-	-	F_{MAX} / 4	-

Table 60. Peripheral clock distribution summary

Peripherals	Clock types	Clock sources	Kernel clock MUX		Max Kernel clock freq. [MHz] (1)	Type (2)(3)
			Pos.	Control field		
LPTIM4, LPTIM5	Kernel	pclk4	**0**(4)	LPTIM45SEL	F_{MAX} / 4	A
		pll2_p_ck	1			
		pll3_r_ck	2			
		lse_ck	3			
		lsi_ck	4			
		per_ck	5			
	Bus	pclk4	-	-	F_{MAX} / 4	-
LPUART1	Kernel	pclk4	**0**(4)	LPUART1SEL	F_{MAX} / 4	A
		pll2_q_ck	1			
		pll3_q_ck	2			
		hsi_ker_ck	3			
		csi_ker_ck	4			
		lse_ck	5			
	Bus	pclk4	-	-	F_{MAX} / 4	-
NPU	Kernel	sys_cpu_ck	-	-	F_{MAX}	S
	Bus(5)	aclk	-		F_{MAX} / 2	-
		hclk1	-			
PWR	Bus	hclk4	-	-	F_{MAX} / 2	-
XSPI1	Kernel	hclk5	**0**(4)	XSPI1SEL	F_{MAX} / 2	A
		pll2_s_ck	1			
		pll2_t_ck	2			
	Bus	hclk5	-	-	F_{MAX} / 2	-
		aclk				
OTGFS	Kernel	hsi48_ker_ck	**0**(4)	OTGFSSEL	50	A
		pll3_q_ck	1			
		hse_ker_ck	2			
	Bus	hclk1	-	-	F_{MAX} / 2	-
RCC	Bus	hclk4	-	-	F_{MAX} / 2	-
RNG	Kernel	hsi48_ker_ck	-	-	48	A
	Bus	hclk3	-	-	F_{MAX} / 2	-

Table 60. Peripheral clock distribution summary

Peripherals	Clock types	Clock sources	Kernel clock MUX		Max Kernel clock freq. [MHz] (1)	Type (2)(3)
			Pos.	Control field		
RTC/AWU(6)	Kernel	no clock	**0**(4)	RTCSEL	1	A
		lse_ck	1			
		lsi_ck	2			
		hse_ker_ck / (RTCDIV+1)	3			
	Bus	pclk4	-	-	F_{MAX} / 4	-
SBS	Bus	pclk4	-	-	F_{MAX} / 4	-
SDMMC1	Kernel	pll2_s_ck	**0**(4)	SDMMC12SEL	200	A
		pll2_t_ck	1			
	Bus	hclk5	-	-	F_{MAX} / 2	-
SPI6	Kernel	pclk4	**0**(4)	SPI6SEL	F_{MAX} / 2	A
		pll2_q_ck	1			
		pll3_q_ck	2			
		hsi_ker_ck	3			
		csi_ker_ck	4			
		hse_ker_ck	5			
	Bus	pclk4	-	-	F_{MAX} / 4	-
SPI/I2S1	Kernel	pll1_q_ck	**0**(4)	SPI1SEL	F_{MAX} / 2	A
		pll2_p_ck	1			
		pll3_p_ck	2			
		AUDIOCLK	3			
		per_ck	4			
	Bus	pclk2	-	-	F_{MAX} / 2	-
SPI4, SPI5	Kernel	pclk2	**0**(4)	SPI45SEL	F_{MAX} / 2	A
		pll2_q_ck	1			
		pll3_q_ck	2			
		hsi_ker_ck	3			
		csi_ker_ck	4			
		hse_ker_ck	5			
	Bus	pclk2	-	-	F_{MAX} / 2	-

Table 60. Peripheral clock distribution summary

Peripherals	Clock types	Clock sources	Kernel clock MUX		Max Kernel clock freq. [MHz] (1)	Type (2)(3)
			Pos.	Control field		
SPI/I2S3, SPI/I2S2	Kernel	pll1_q_ck	**0**(4)	SPI23SEL	F_{MAX} / 2	A
		pll2_p_ck	1			
		pll3_p_ck	2			
		AUDIOCLK	3			
		per_ck	4			
	Bus	pclk1	-	-	F_{MAX} / 4	-
TAMPER	Kernel	no clock	**0**(4)	RTCSEL	4	A
		lse_ck	1			
		lsi_ck	2			
		hse_ker_ck/ (RTCDIV+1)	3			
	Bus	pclk4	-	-	F_{MAX} / 4	-
DTS	Kernel	lse_ck	-	-	10	A
	Bus	pclk4	-	-	F_{MAX} / 4	-
TIM2, TIM3, TIM4, TIM5, TIM6, TIM7, TIM10, TIM11, TIM12, TIM13, TIM14	Kernel	tim_ck	-	-	F_{MAX}	S
	Bus	pclk1	-	-	F_{MAX} / 4	-
TIM1, TIM8, TIM9, TIM15, TIM16, TIM17, TIM20	Kernel	tim_ck	-	-	F_{MAX}	S
	Bus	pclk2	-	-	F_{MAX} / 2	-
USART1, USART6, UART9, USART10	Kernel	pclk2	**0**(4)	USART16910SEL	F_{MAX} / 4	A
		pll2_q_ck	1			
		pll3_q_ck	2			
		hsi_ker_ck	3			
		csi_ker_ck	4			
		lse_ck	5			
	Bus	pclk2	-	-	F_{MAX} / 2	-
USART2, USART3, UART4, UART5, UART7, UART8	Kernel	pclk1	**0**(4)	UART234578SEL	F_{MAX} / 4	A
		pll2_q_ck	1			
		pll3_q_ck	2			
		hsi_ker_ck	3			
		csi_ker_ck	4			
		lse_ck	5			
	Bus	pclk1	-	-	F_{MAX} / 4	-

Table 60. Peripheral clock distribution summary

Peripherals	Clock types	Clock sources	Kernel clock MUX		Max Kernel clock freq. [MHz] (1)	Type (2)(3)
			Pos.	Control field		
UCPD	Kernel	ucpd_ker_ck	-	-	25	A
	Bus	pclk1	-	-	F_{MAX} / 4	-
VREFBUF	Bus	pclk4	-	-	F_{MAX} / 4	-
WWDG1	Bus	pclk1	-	-	F_{MAX} / 4	-

1. The F_{MAX} value depends on the device reference and can be found on the datasheet of the product.
2. 'A' Means that the kernel clock is asynchronous with respect to the bus interface clock.
3. 'S' Means that the kernel clock is synchronous with respect to the bus interface clock.
4. Reset value
5. NPU bus clocks are in reality clock enable signals, which indicate the NPU core (kernel) clock cycle on which the bus is valid
6. The RTC switch is in the V_{SW} voltage domain

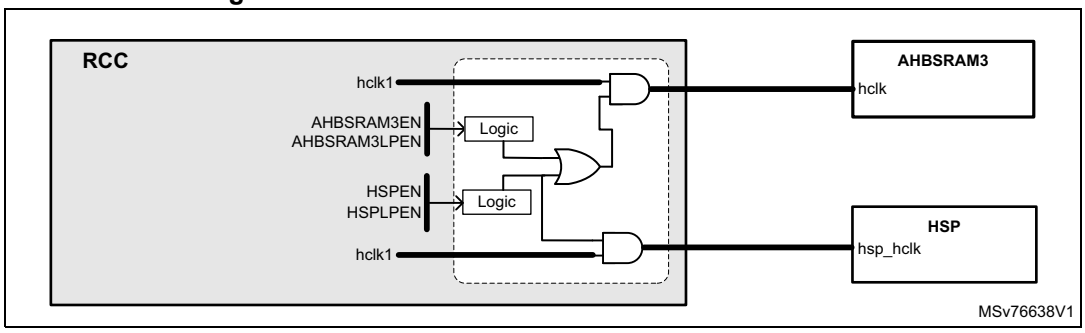
Figure 50 to *Figure 65* provide a more detailed description of kernel clock distribution. Refer to *Section 7.5.12: Peripheral clock gating control* for more details.

To reduce the number of switches, some peripherals share the same kernel clock source. Nevertheless, all peripherals have their dedicated enable signal.

Clock distribution for HSP

The HSP is clocked by the AHB1 peripheral clock, as shown in *Figure 50*.

Figure 50. Clock distribution for HSP and AHBSRAM3



The HSP can operate in Sleep mode, when the HSPLPEN bit is set to 1, but not in Stop mode. The HSP uses AHBSRAM3, so whenever the HSP clock is enabled, the AHBSRAM3 clock is also enabled.

Clock distribution for ADF

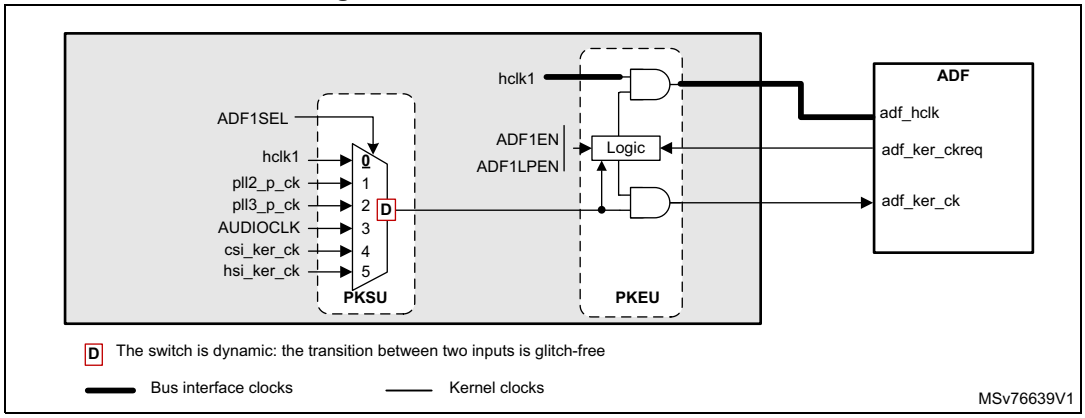
The audio peripherals generally need specific accurate frequencies. As shown in *Figure 51*, the kernel clock of the ADF can be generated by:

- AHB peripheral clock
- PLL2 or 3 for optimal flexibility in frequency generation
- HSI or CSI for use-cases where the current consumption is critical
- AUDIOCLK when an external clock reference needs to be used

The ADF can work even in Stop mode as long as the bus clock is not requested. Working in Stop mode is possible only if the application selected csi_ker_ck or hsi_ker_ck as kernel clock sources. The ADF asserts the adf_ker_ckreq when the kernel clock is requested.

The kernel clock mechanism works only if the kernel clock source is provided by an RC oscillator (**hsi_ker_ck** or **csi_ker_ck**).

Figure 51. Clock distribution for ADF



1. **X** represents the selected mux input after a system reset.

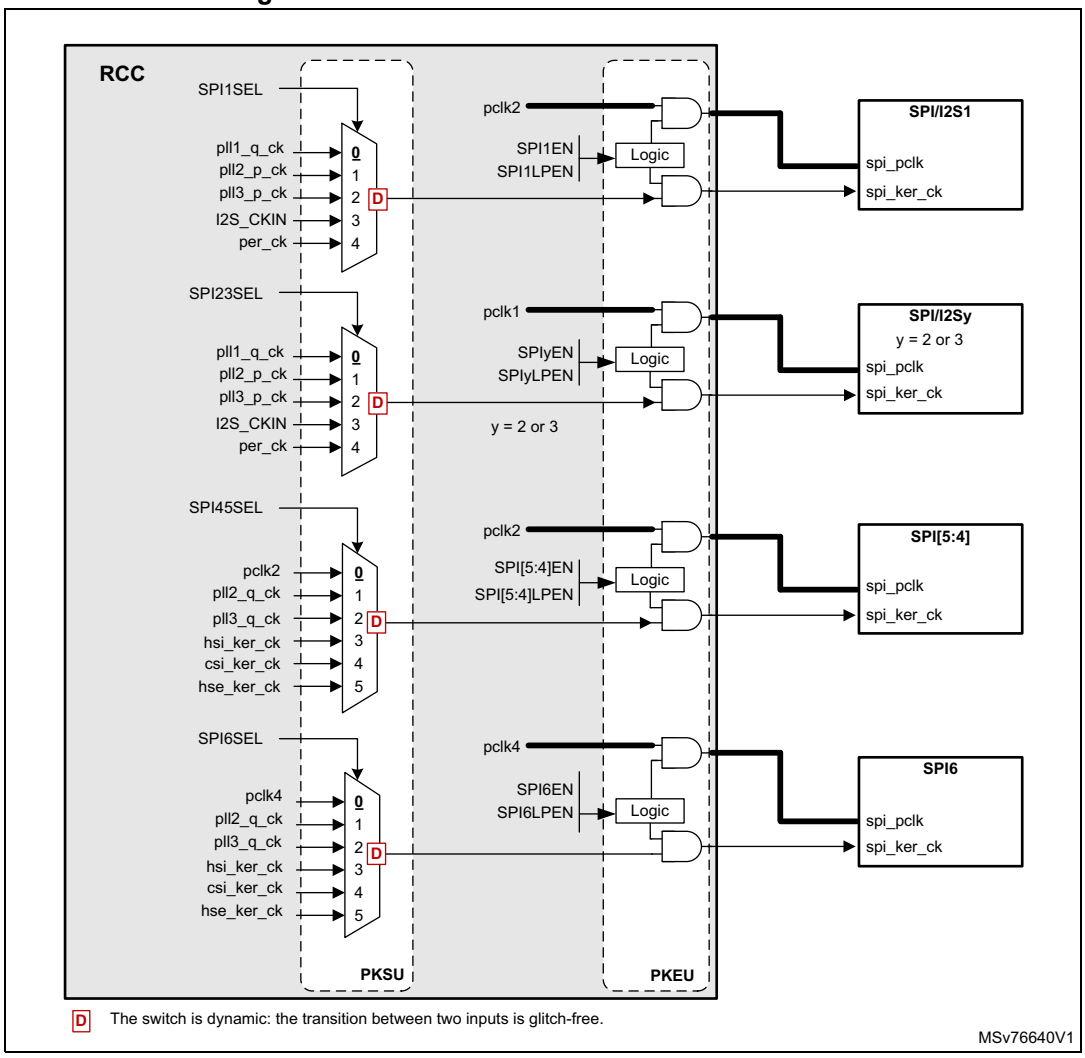
Clock distribution for SPIs, SPI/I2Ss

SPI peripherals do not need an accurate kernel clock frequency but a clock fast enough for the serial interface. The SPI/I2S peripherals may request an accurate kernel clock frequency when using the I2S function. For that purpose the source can be selected among the following ones:

- The bus clock (for SPIs)
- PLL1 when the number of active PLLs must be reduced
- PLL2 or PLL3 if better flexibility is required. As an example, this solution allows changing the frequency bus via PLL1 without affecting the speed of some serial interfaces.
- HSI or CSI or per_ck for low-power use-cases.

The SPI/I2S also have common kernel clocks with the ADF in order to ease data exchange.

Figure 52. Clock distribution for SPIs and SPI/I2S



1. **X** represents the selected switch input after a system reset.

Clock distribution for I2C[3:1], and I3C1

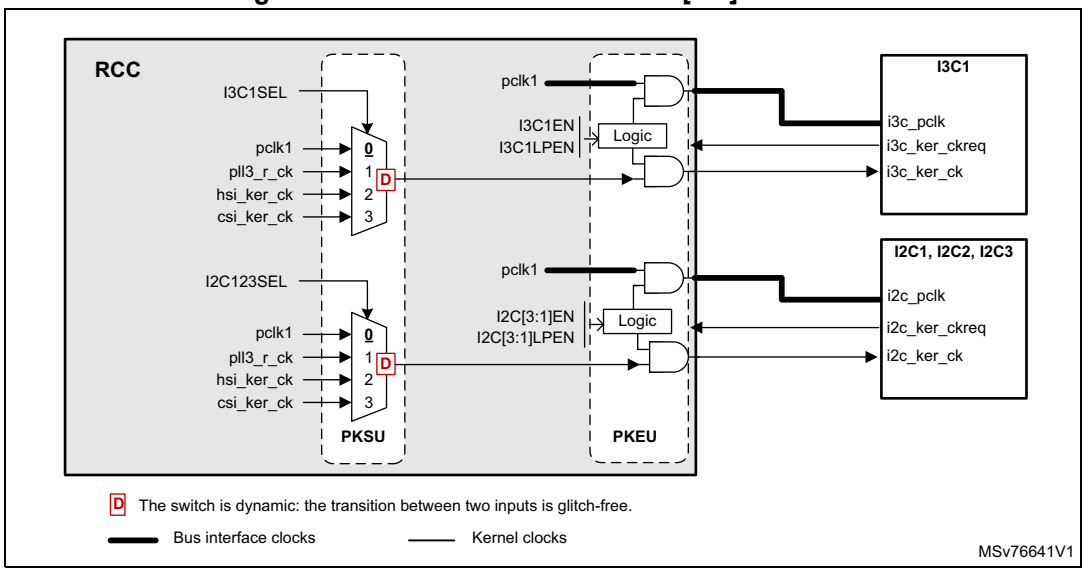
I2C and I3C peripherals do not need an accurate kernel clock frequency but a clock fast enough to properly handle the serial interface. The following kernel clock sources are proposed:

- The bus clock
- PLL3 in the case where the bus clock cannot be used
- HSI or CSI for low-power use-cases

The I2Cs and I3C1 can work even in Stop mode as long as the bus clock is not requested. Working in Stop mode is possible only if the application selected **csi_ker_ck** or **hsi_ker_ck** as kernel clock sources. The I2Cs and I3C1 assert the **i2c_ker_ckreq** or **i3c_ker_ckreq** signals, when the kernel clock is requested.

The kernel clock mechanism works only if the kernel clock source is provided by an RC oscillator (**hsi_ker_ck** or **csi_ker_ck**).

Figure 53. Clock distribution for I2C[3:1] and I3C1



1. **X** represents the selected switch input after a system reset

Clock distribution for UARTs and USARTs

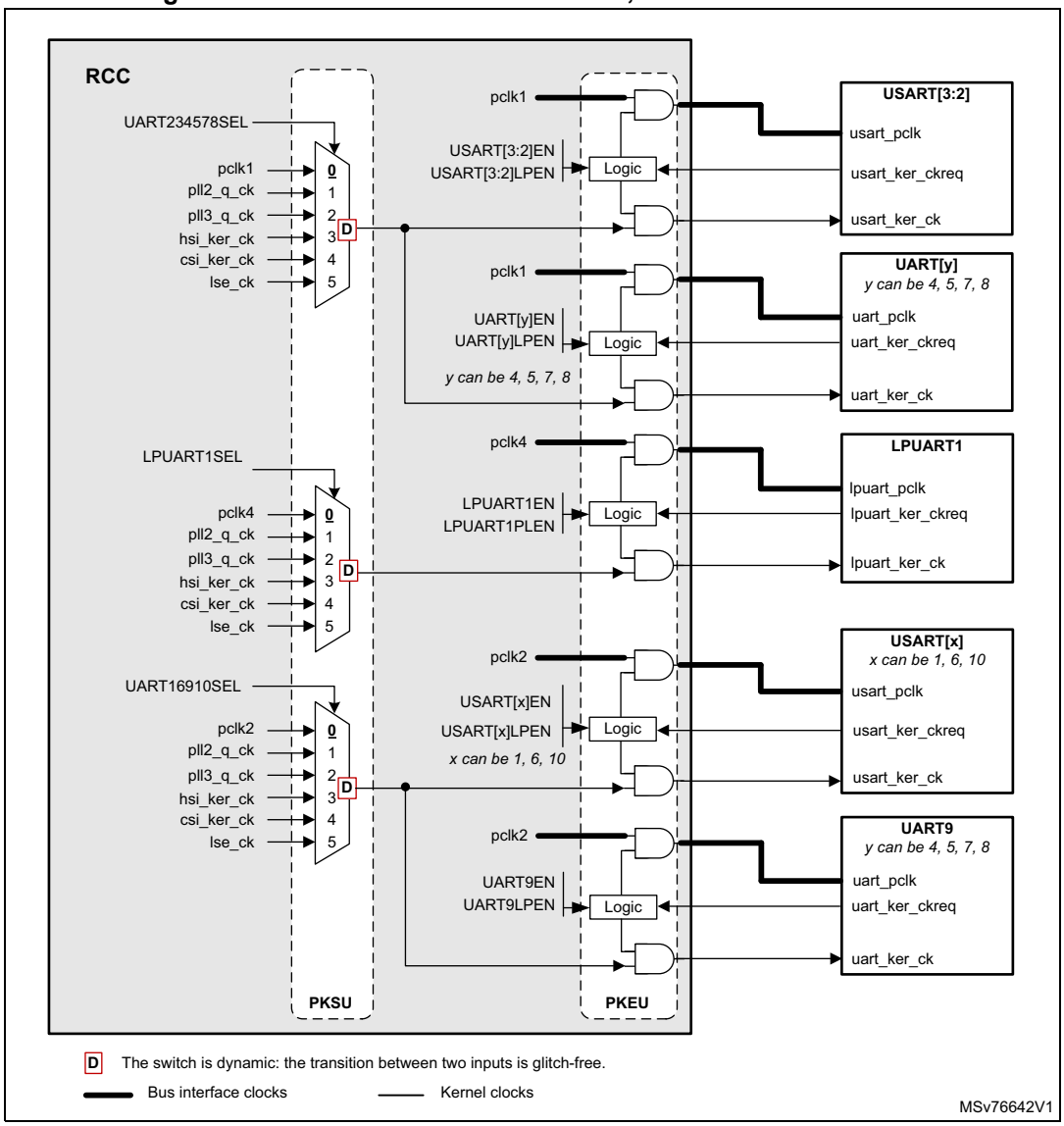
UARTs need a kernel clock frequency allowing the adjustment of the baud rate with an acceptable accuracy (generally less than ±2%). For that purpose the source can be selected among the following ones:

- The bus clock
- PLL3 in the case where the bus clock cannot be used
- HSI, CSI or LSE for low-power use-cases

The UARTs can work even in Stop mode as long as the bus clock is not requested. Working in Stop mode is possible only if the application selected csi_ker_ck or hsi_ker_ck as kernel clock sources. The UARTs assert the **uartx_ker_ckreq** signal, when the kernel clock is requested. The kernel clock mechanism works only if the kernel clock source is provided by an RC oscillator (**hsi_ker_ck** or **csi_ker_ck**).

UARTs can also work in Stop mode using the LSE clock when high baud rates are not required.

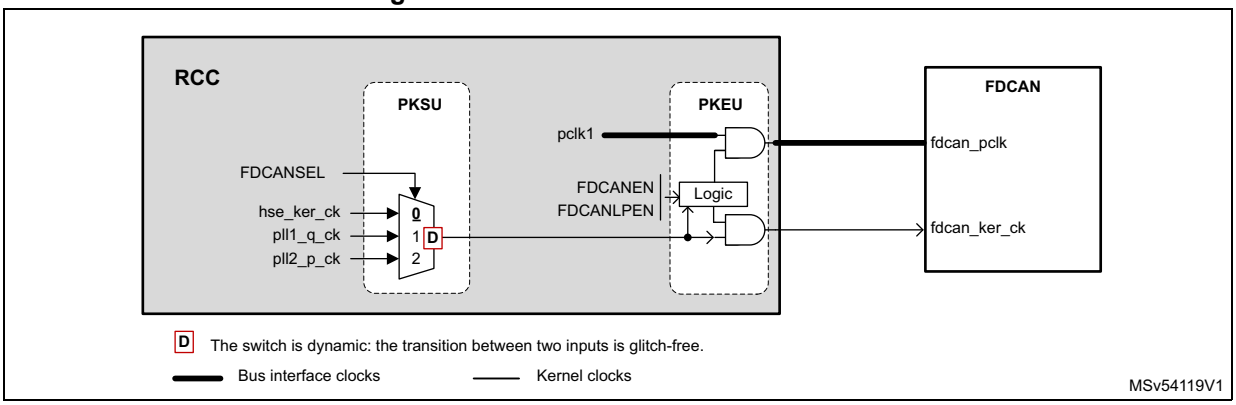
Figure 54. Clock distribution for UARTs, LPUART1 and USARTs



1. **X** represents the selected switch input after a system reset.

Clock distribution for FDCAN

Figure 55. Clock distribution for FDCAN



1. **X** represents the selected switch input after a system reset.

Clock distribution for SDMMC, FMC, and XSPI

The FMC kernel clock can be chosen between four different sources, giving good flexibility.

For XSPI, a clock switch allows selection between three different sources, giving good flexibility.

The switches FMCSEL, XSPISEL, also embeds several protections:

- In case of HSE failure detected by the HSECSS function, the XSPISEL switch and the FMCSEL switch go to recovery position in order to provide a default clock.
- A protection bit: FMCCKP prevents the application to accidentally change the FMCSEL value. Refer to *Section 7.5.7: Clock protection* for details.
- A protection bit: XSPICKP prevents the application to accidentally change the XSPISEL value. Refer to *Section 7.5.7: Clock protection* for details.
- Status fields (FMCSWP, XSPISWP) located in the *RCC clock protection register (RCC_CKPROTR)* allow the application to check the effective position of the switches.
- If one of the switches is selecting an input with an invalid clock, the switch outputs a recovery clock preventing a memory crash and allowing the application to log or correct the issue. The RCC provides **hclk5**/4 as recovery clock. In order to work fine, the application must ensure that the recovery clock frequency is supported by the external device.
- For the XSPISEL switches:
  - Going to position 1 is allowed if PLL2ON = PLL2SEN = 1. If the application switches to position 1 when these conditions are not met, the switch generates the recovery clock.
  - Going to position 2 is allowed if PLL2ON = PLL2TEN = 1. If the application switches to position 2 when these conditions are not met, the switch generates the recovery clock.
- For the FMCSEL switch:
  - Going to position 1 is allowed if PLL1ON = PLL1QEN = 1. If the application switches to position 1 when these conditions are not met, the switch generates the recovery clock.
  - Going to position 2 is allowed if PLL2ON = PLL2REN = 1. If the application switches to position 2 when these conditions are not met, the switch generates the

recovery clock.

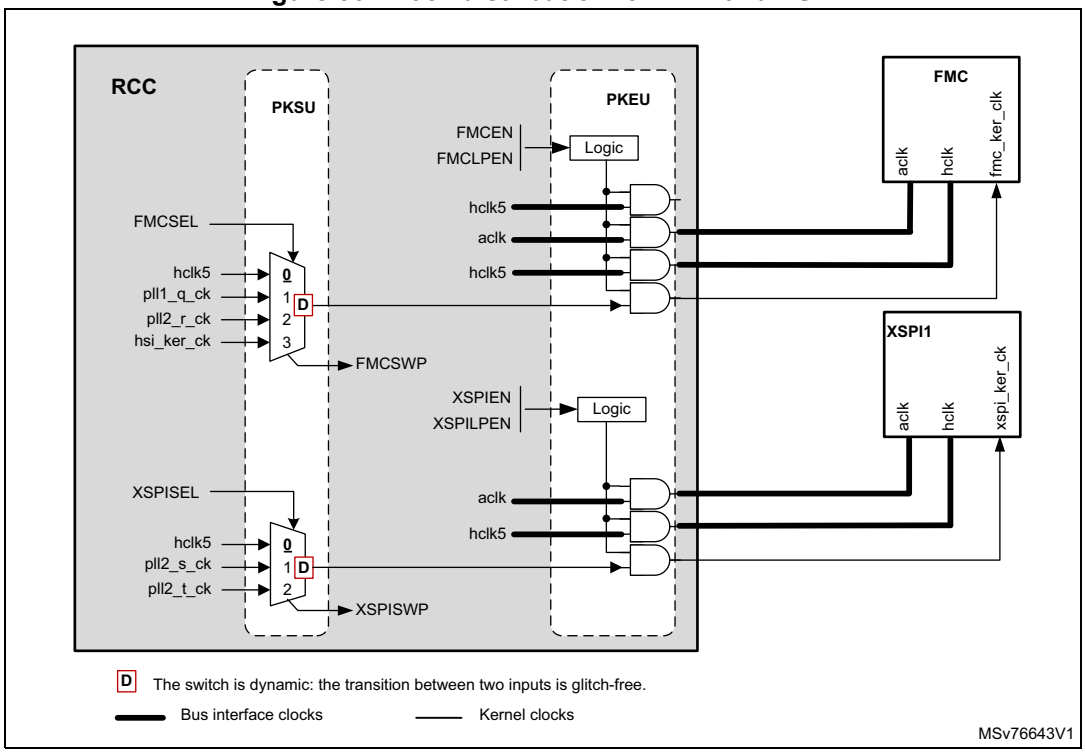
– Going to position 3 is allowed if HSION = 1. If the application switches to position 3 when this condition is not met, the switch generates the recovery clock.

How to handle XSPI switches properly:

- Switch on the wanted clock source,
- Ensure the clock source is ready, and the conditions described above are met,
- Set XSPISEL to the wanted position,
- Enable the XSPI clock (XSPIEN = 1),
- Read XSPISWP field from RCC_CKPROTR register,
- If the position given by XSPISWP is the same as the wanted, it means that the XSPI kernel clock is as expected.
- If the position given by XSPISWP is 000, it means that there is no clock present at the selected input.

The XSPI must provide a clock to the external memory, with a duty cycle distortion generally lower than 5%. In order to reach this requirement, the kernel clock provided to the XSPI has a typical duty cycle of 50%. In addition, the XSPI embeds a prescaler allowing clock division by even ratios.

Figure 56. Clock distribution for FMC and XSPI



1. **X** represents the selected switch input after a system reset.

The SDMMC1 and SDMMC2 share the same kernel clock. A clock switch allows the selection between two different sources. It is possible to enable independently each SDMMC block. Note that when one of the SDMMC is enabled via its SDMMC[2:1]EN bit, the associated delay block is enabled as well.

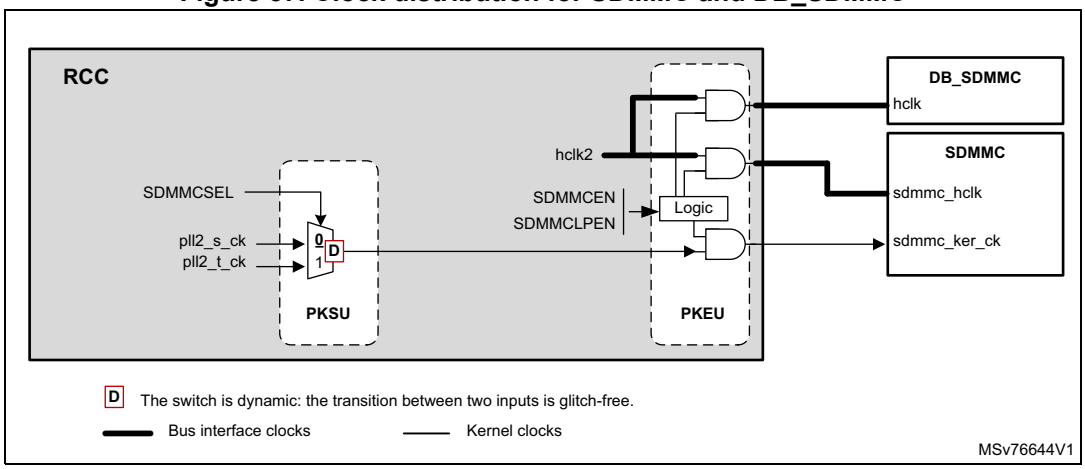
The application must take care about the duty-cycle of the kernel clock provided to the SDMMC blocks.

Table 61. SDMMC interface clock constraints

SDMMC mode	Mode name	Interface clock frequency	Duty cycle constraint
SDIO	SDR12	25 MHz or less	30 - 70%
	SDR25	50 MHz or less	30 - 70%
	DDR50	50 MHz or less	45 - 55%
	SDR50	100 MHz or less	30 - 70%
e.MMC	Backward compatible	26 MHz or less	30 - 70%
	High Speed SDR	52 MHz or less	30 - 70%
	High Speed DDR	52 MHz or less	45 - 55%

For example, if the SDMMC works in SDR50, then a kernel clock of 50 MHz, with a duty cycle better than 30-70% is enough. If the SDMMC works in DDR50, then it is recommended to provide a kernel clock of 100 MHz, and divide the frequency of the kernel clock by two, using the SDMMC divider in order to insure a duty-cycle very close to 50% for the SDMMC_CK.

Figure 57. Clock distribution for SDMMC and DB_SDMMC



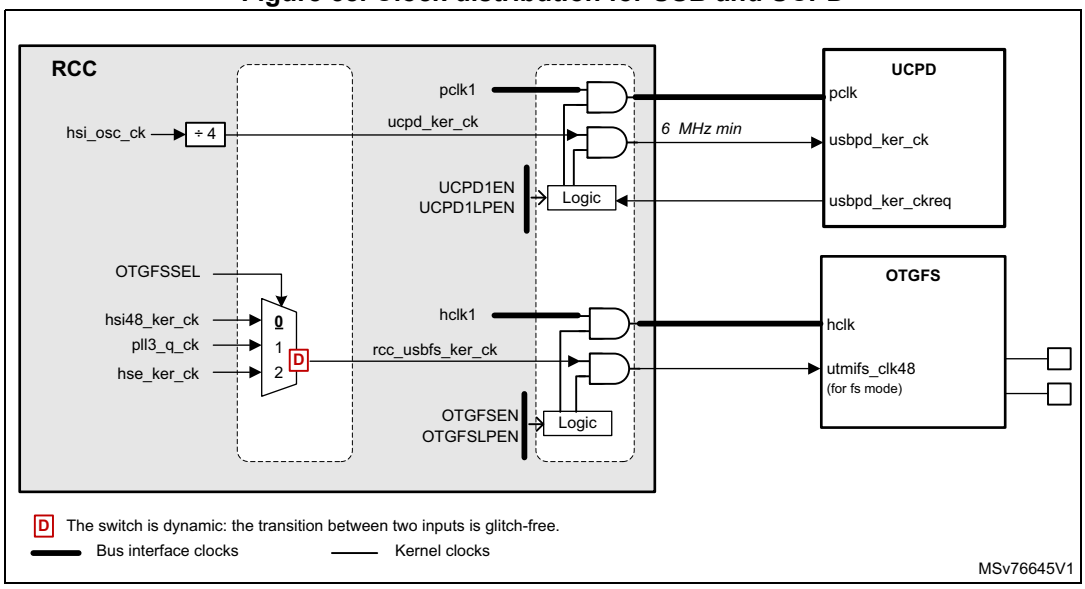
1. **X** represents the selected switch input after a system reset.

Clock distribution for OTGFS and UCPD

Figure 58 shows the clock distribution for the USB blocks.

- The USB type-C power delivery block (UCPD), uses ucpd_ker_ck as kernel clock. The ucpd_ker_ck is directly generated from the HSI output divided by 4.
- The OTGFS receives a 48 MHz clock. The application can select:
  - The hsi48_ker_ck for crystal-less applications
  - hse_ker_ck or
  - pll3_q_ck

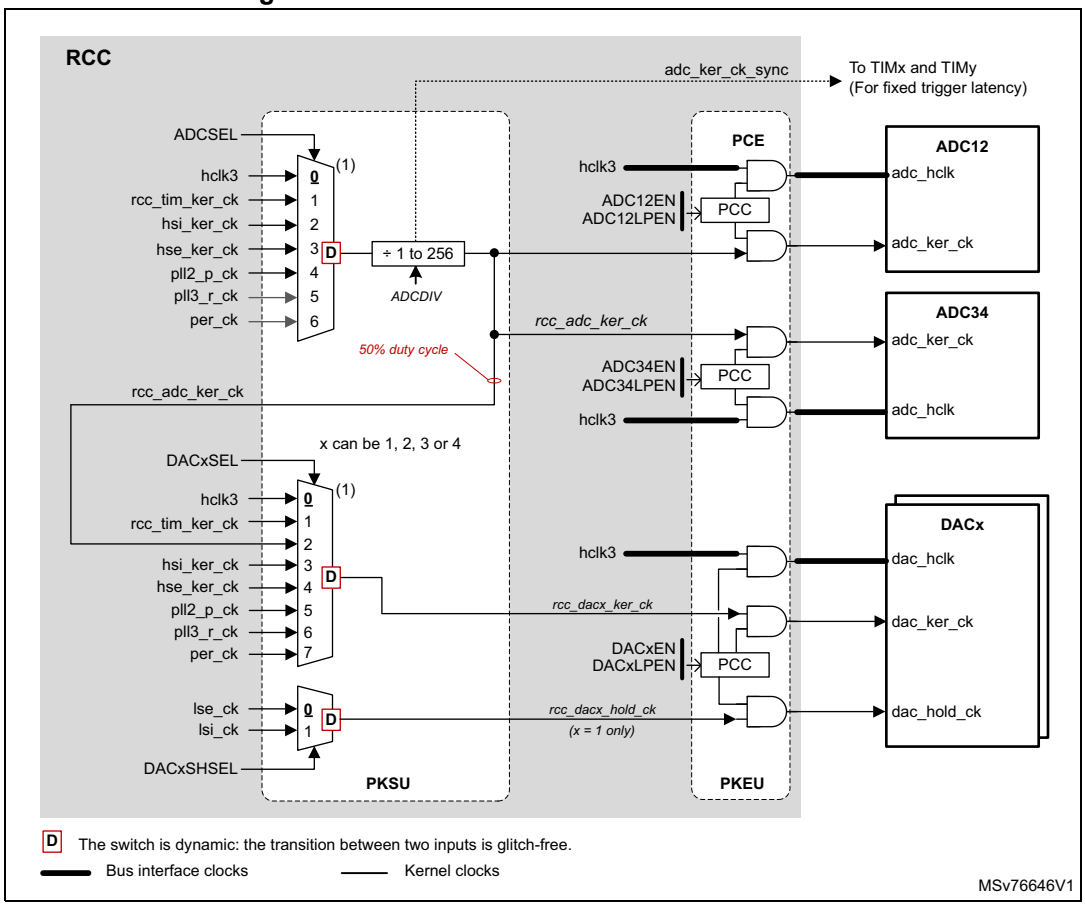
Figure 58. Clock distribution for USB and UCPD



1. **X** represents the selected switch input after a system reset.

Clock distribution for ADCs and DACs

Figure 59. Clock distribution for ADCs and DACs



1. **X** represents the selected switch input after a system reset.

Every ADC receives the same kernel clock, rcc_adc_ker_ck, to minimize interference. The ADC kernel clock can be selected from a variety of sources:

- The bus clock (hclk3) - can be set to Fcpu or Fcpu/2
- The timers kernel clock (rcc_tim_ker_ck) - Fcpu only
- The HSI or HSE oscillator clocks
- The PLL2 P divider or PLL3 R divider outputs
- The peripheral clock (per_ck)

To synchronize the ADCs with the Timers, select hclk3 or rcc_tim_ker_ck as the ADC kernel clock.

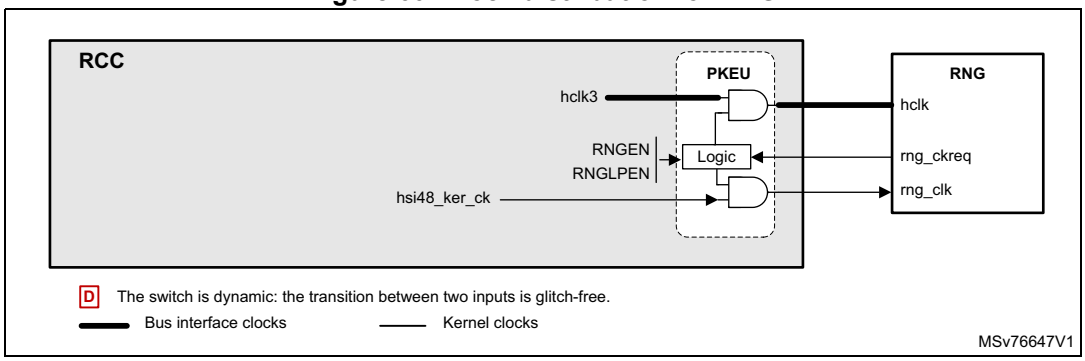
Similarly, the DACs all receive the same kernel clock, rcc_dac_ker_ck. The same choice of sources is available for the DAC kernel clock. In addition, the ADC kernel clock can be selected for the DACs as well. This can help reduce interference between DAC and ADC.

If the bus clock, hclk3, is selected as the DAC kernel clock, the DACs are automatically switched to synchronous mode, which reduces the conversion delay.

The DAC sample and hold clock (rcc_dacx_hold_ck) is used to refresh the DAC output voltage if there is a slow sample rate (< 32 kHz). It can be selected from either the LSE (lse_ck) or the LSI (lsi_ck) outputs. This only applies to buffered DACs (that is DAC1).

Clock distribution for RNG

Figure 60. Clock distribution for RNG



1. **X** represents the selected switch input after a system reset.

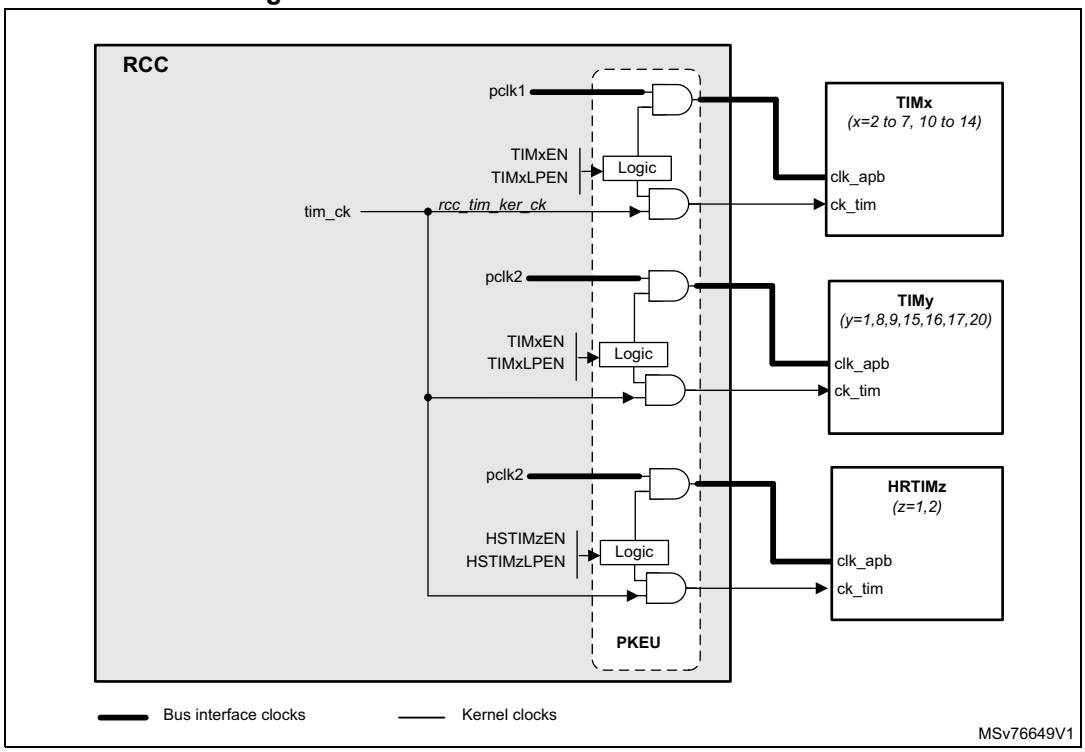
The RNG block uses the hsi48_ker_ck as kernel clock. The logic handling controlling the clock of the RNG is different from standard peripherals. When enabled, the RNG has to assert its kernel clock request in order to get a kernel clock. Refer to *Section 7.5.12: Peripheral clock gating control* for details.

Clock distribution for HRTIM, TIM and LPTIM

The timers use kernel clocks (tim_ck) phase aligned with the bus interface clock.

The working frequency of the timers clock is equal to the CPU clock frequency.

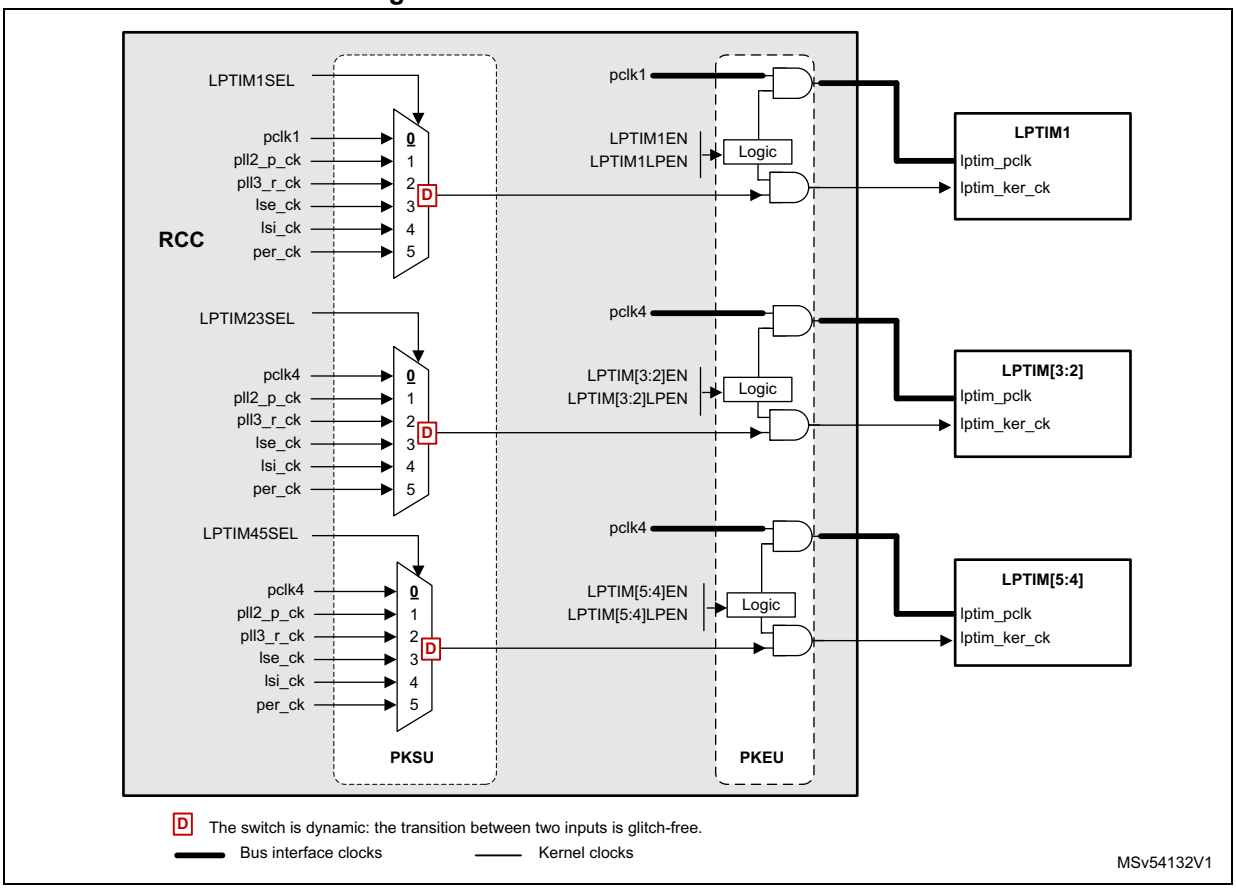
Figure 61. Clock distribution for TIM and HRTIM



The LPTIM timers have a kernel clock fully asynchronous with respect to their bus interface clock. The kernel clock can be selected among up to six clock sources.

The LPTIMs also use lse_ck and lsi_ck as kernel clock, allowing them to work even when the system is in Stop mode.

Figure 62. Clock distribution for LPTIMs



1. **X** represents the selected switch input after a system reset

Clock distribution for RTC/AWU clock

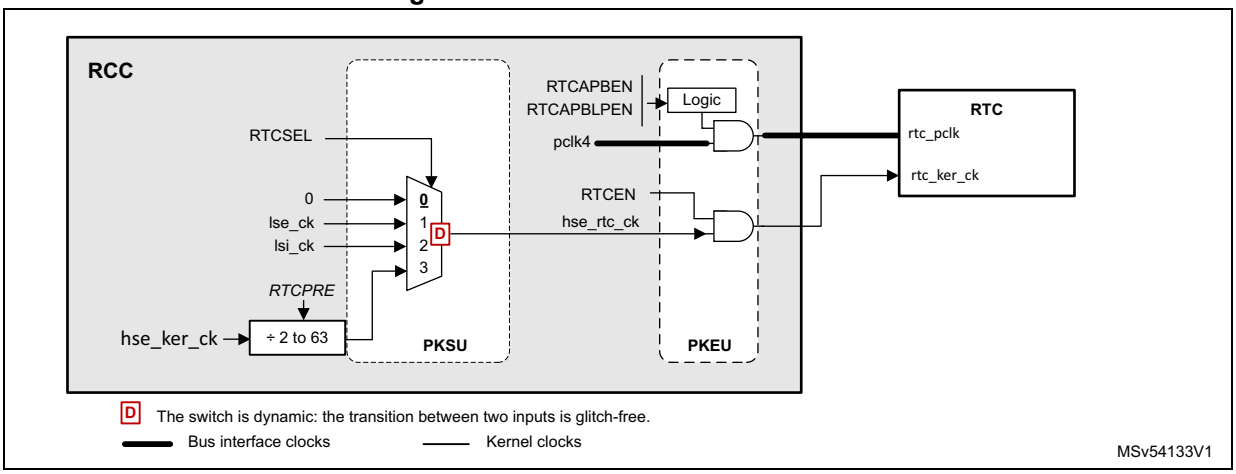
The **rtc_ck** clock source can be one of the following:

- The **hse_rtc_ck** (**hse_ker_ck** divided by a programmable prescaler)
- The **lse_ck**
- The **lsi_ck** clock

The source clock is selected by programming the RTCSEL[1:0] bits in the *RCC Backup domain control register (RCC_BDCR)* and the RTCPRE[5:0] bits in the *RCC clock configuration register (RCC_CFGR)*.

This selection cannot be modified without resetting the Backup domain.

Figure 63. Clock distribution for RTC



1. **X** represents the selected switch input after a system reset

If the LSE is selected as RTC clock, the RTC works normally even if the backup or the V_{DD} supply disappears.

The LSE clock is in the Backup domain, whereas the other oscillators are not. As a consequence:

- If LSE is selected as RTC clock, the RTC continues working even if the V_{DD} supply is switched OFF, provided the V_{BAT} supply is maintained.
- If LSI is selected as the RTC clock, the AWU state is not guaranteed if the V_{DD} supply is powered off.
- If the HSE clock is used as RTC clock, the RTC state is not guaranteed if the V_{DD} supply is powered off or if the V_{CORE} supply is powered off. In addition the HSE is not available if the system goes to Stop.

The **rtc_ck** clock is enabled through the RTCEN bit located in the *RCC Backup domain control register (RCC_BDCR)*.

The RTC bus interface clock (APB clock) is enabled through RTCAPBEN and RTCAPBLPEN bits located in RCC_APB4ENR/LPENR registers.

Note: To read the RTC calendar register when the APB clock frequency is less than seven times the RTC clock frequency ($F_{APB} < 7 \times F_{RTCLCK}$), the software must read the calendar time and date registers twice. The data are correct if the second read access to RTC_TR gives the same result than the first one. Otherwise a third read access must be performed.

Clock distribution for watchdog

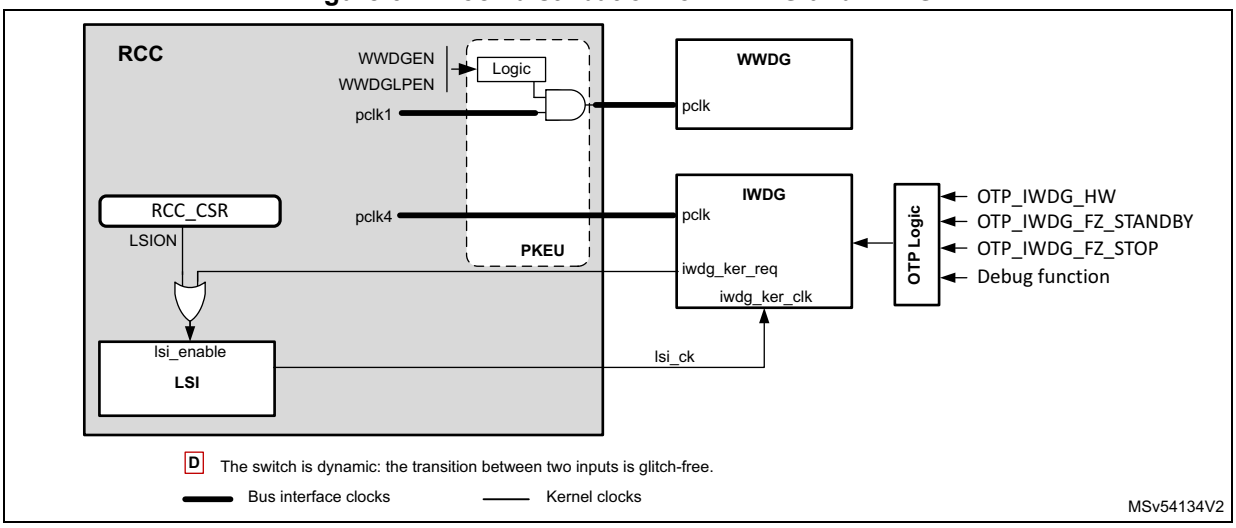
The RCC provides the clock for the two watchdog blocks available on the circuit. The independent watchdog (IWDG) is connected to the LSI. The window watchdog (WWDG) is connected to the APB clock.

If an independent watchdog is started by either hardware option or software access, the LSI is forced ON and cannot be disabled. After the LSI oscillator setup delay, the clock is provided to the IWDG.

The window watchdog clock (pclk1) can be enabled by setting the WWDGEN bit in the RCC_APB1ENR register. The software cannot stop WWDG down-counting by setting the WWDGEN bit to 0.

The WWDG block is frozen when the MCU goes to Stop mode.

Figure 64. Clock distribution for WWDG and IWDG



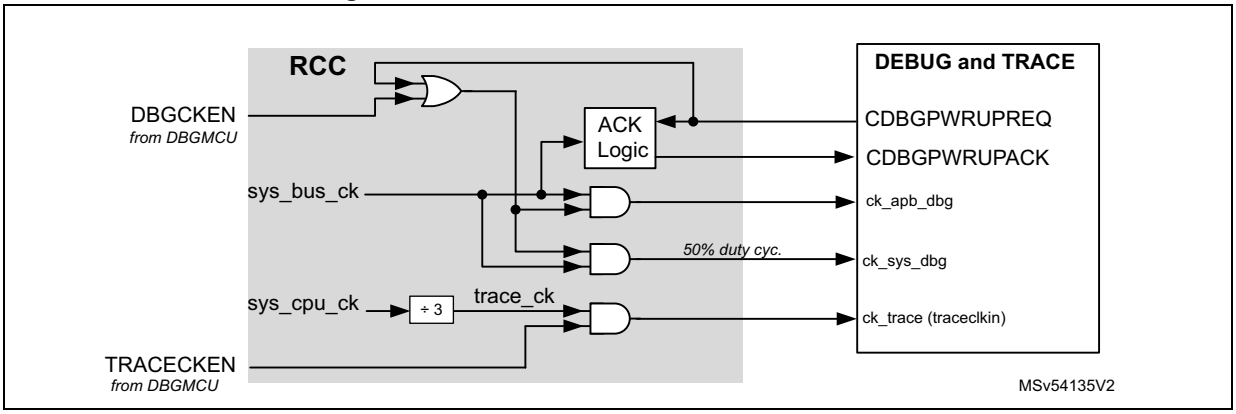
1. **X** represents the selected switch input after a system reset

Clock distribution for Debug and Trace

The clock generation for the trace and debug is controlled by the DBGMCU block. The DBGCKEN bit allows the application to provide a clock to the debug components. It is also possible to enable this clock via the Debug Access Port.

The trace clock generation is controlled via TRACECKEN bit.

Figure 65. Clock distribution for DBG and trace



1. **X** represents the selected switch input after a system reset

Clock frequency measurement using TIMx

Most of the clock source generator frequencies can be measured by means of the input capture of TIMx.

- Calibrating the HSI or CSI with the LSE:

  The primary purpose of having the LSE connected to a TIMx input capture is to be able to accurately measure the HSI or CSI. This requires to use the HSI or CSI as system clock source either directly or via PLL1. The number of system clock counts between

consecutive edges of the LSE signal gives a measurement of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppm) we can determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process and/or temperature- and voltage-related frequency deviations.

The basic concept consists in providing a relative measurement (e.g. HSI/LSE ratio). The precision is therefore tightly linked to the ratio between the two clock sources. The greater the ratio is, the more accurate the measurement is.

The HSI and CSI oscillators have dedicated user-accessible calibration bits for this purpose (see *RCC CSI calibration register (RCC_CSICFGR)*). When HSI or CSI is used via the PLLx, the system clock can also be fine-tuned by using the fractional divider of the PLLs.

- Calibrating the LSI with the HSI:

  The LSI frequency can also be measured: this is useful for applications that do not have a crystal. The ultra-low-power LSI oscillator has a large manufacturing process deviation. The LSI clock frequency can be measured using the more precise HSI clock source. Using this measurement, a more accurate RTC time base timeouts (when LSI is used as the RTC clock source) and/or an IWDG timeout with an acceptable accuracy can be obtained.

7.5.10 General clock concept overview

The RCC handles the distribution of the CPU, bus interface and peripheral clocks for the system, according to the CPU operating mode (refer to *Section 7.5.1: Clock naming convention* for details on clock definitions).

For each peripheral, the application can control the activation/deactivation of its kernel and bus interface clock. Prior to using a peripheral, the CPU must enable it (by setting PERxEN to 1), and define if this peripheral remains active in Sleep mode (by setting PERxLPEN to 1). This is called ‘allocation’ of a peripheral by the CPU (refer to *Section 7.5.11: Peripheral allocation* for more details).

The peripheral allocation is used:

- By the RCC to automatically control the clock gating according to the CPU modes, and
- By the PWR to control the supply voltages of VCORE.

Memory handling

The CPU can access all the memory areas available in the product:

- AXISRAM1, AXISRAM2, AXISRAM3, AXISRAM4, ITCM, DTCM1, DTCM2 and FLASH
- AHBSRAM1, AHBSRAM2 and AHBSRAM3
- BKPRAM

The BKPRAM, AHBSRAM1, AHBSRAM2 and AHBSRAM3 have dedicated enable bits in order to gate the bus interface clock. These enable bits are set to 1 by default, and can be programmed with 0 by software to reduce power consumption if the memories are not actively being used.

The AHBSRAM3 is also enabled when the HSP enable bit is set.

Note: The memory interface clocks (flash memory and RAM interfaces) can be stopped by software during Sleep mode (via the respective LPEN bits).

Refer to *Section 7.5.12: Peripheral clock gating control* and *Section 7.5.13: CPU and bus matrix clock gating control* sections for details on clock enabling.

7.5.11 Peripheral allocation

The CPU can allocate a peripheral and hence control its kernel and bus interface clock.

The CPU can allocate a peripheral by setting the dedicated PERxEN bit to 1.

The CPU can control the peripheral clock gating when it is in Sleep mode via the PERxLPEN bits.

The peripheral allocation bits (PERxEN bits) are used by the hardware to provide the kernel and bus interface clocks to the peripherals. However they are also used to link peripherals to the CPU. In this way, the hardware is able to safely gate the peripheral clocks and bus matrix clocks according to CPU states.

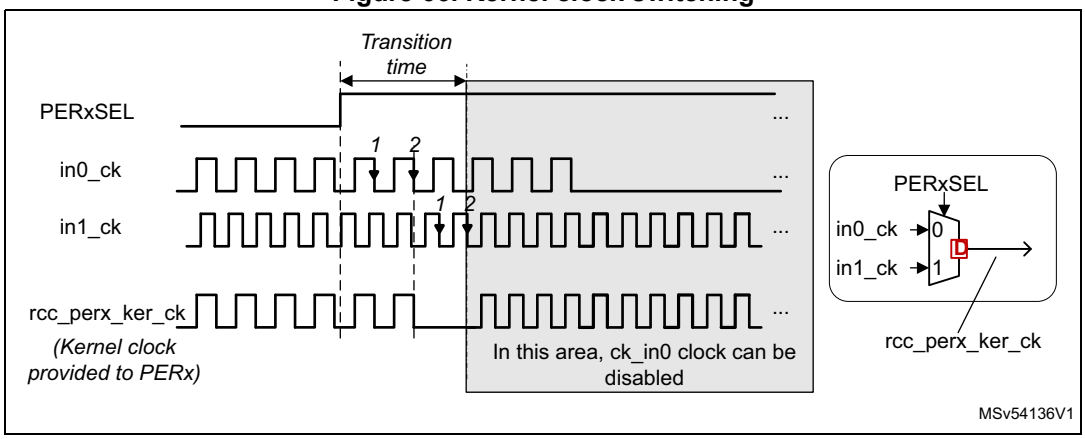
Clock switches and gating

- Clock switching delays

  The input selected by the clock switches can be changed dynamically without generating spurs or timing violation. For example, if PERxSEL (In *Figure 66*) goes from 0 to 1, the switch first disables the clock output using the currently selected clock (in0_ck), and enables again the clock output using the new selected clock (in1_ck). The disable and enable commands are re-synchronized to their respective clocks. If one of the two clocks is not present, the sequence cannot be completed, and no clock is output. To recover from this situation, the user must either provide a valid clock to in1_ck input or set back PERxSEL to 0.

  During the transition from one input to another, the kernel clock provided to the peripheral is gated, in the worst case, during 2 or 3 clock cycles of the new selected clock. As shown in *Figure 66*, both input clocks must be present during transition time.

Figure 66. Kernel clock switching



- Clock enabling delays

  In the same way, the clock gating logic synchronizes the enable command (coming generally from a kernel clock request or PERxEN bits) with the selected clock, in order to avoid generation of spurs.

  – A maximum delay of two periods of the enabled clock may occur between the enable command and the first rising edge of the clock. The enable command can be the rising edge of the PERxEN bits of RCC_xxxxENR registers, or a kernel clock request asserted by a peripheral.

  – A maximum delay of 1.5 periods of the disabled clock may occur between the disable command and the last falling edge of the clock. The disable command can

be the falling edge of the PERxEN bits of RCC_xxxxENR registers, or a kernel clock request released by a peripheral.

Note: Both the kernel and bus interface clocks are affected by this re-synchronization delay.

7.5.12 Peripheral clock gating control

As mentioned previously, each peripheral requires one or several bus interface clocks, named **rcc_perx_bus_ck** (for peripheral x). These clocks can be an APB, AHB or AXI clock, according to which bus, or buses, the peripheral is connected.

The clocks used as bus interface for peripherals, can be **aclk**, **hclk[5:1]**, **pclk[~~5:~~4], or pclk[2:1],** depending on the bus connected to each peripheral.

Some peripherals also require dedicated clocks for their communication interface. These clocks are generally asynchronous with respect to the bus interface clock. They are named kernel clocks (**perx_ker_ck**). Both bus interface and kernel clocks can be gated according to several conditions detailed hereafter.

As shown in *Figure 67*, enabling the kernel and bus interface clocks of each peripheral depends on several input signals:

- PERxEN and PERxLPEN bits,

  PERxEN represents the peripheral enable (allocation) bit for the CPU. The CPU can write these bits to 1 via the RCC_xxxxENR registers.
- CPU state (**cpu_sleep** and **cpu_deepsleep** signals)
- Kernel clock request (**perx_ker_ckreq**) of the peripheral itself, when the feature is available

Refer to *Section 7.5.11: Peripheral allocation* for more details.

Figure 67. Peripheral kernel clock enable logic details

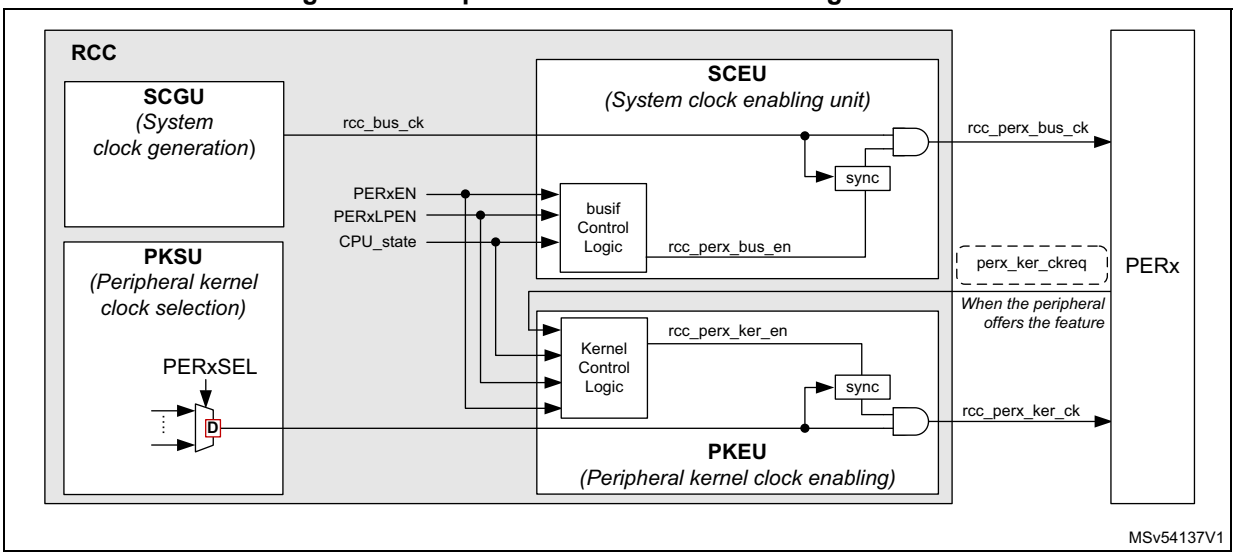


Table 62 gives a detailed description of the enabling logic of the peripheral clocks for peripherals located in the CPU domain and allocated by the CPU.

Table 62. Peripheral clock enabling

PERxEN	PERxLPEN	PERxSEL	perx_ker_ckreq	CPU state	rcc_perx_ker_ck	rcc_perx_bus_ck	Comments
0	X	X	X	X	-	-	No clock provided to the peripheral, because PERxEN=0
1	X	X	X	Run	CK	CK	Kernel and bus interface clocks are provided to the peripheral, because the CPU is in Run, and PERxEN = 1.
1	0	X	X	Sleep	-	-	No clock provided to the peripheral, because the CPU is in Sleep and PERxLPEN = 0
1	1				CK	CK	Kernel and bus interface clocks are provided to the peripheral, because CPU is in Sleep, and PERxLPEN = 1
1	0	X	X	Stop	-	-	No clock provided to the peripheral because the PERxLPEN bit is set to 0.
1	1	no lsi_ck and no lse_ck and no hsi_ker_ck and no csi_ker_ck	X		-	-	No clock provided to the peripheral because CPU is in Stop and **lse_ck** or **lsi_ck** or **hsi_ker_ck** or **csi_ker_ck** are not selected as kernel clock.
1	1	lsi_ck or lse_ck	X		CK	-	Kernel clock is provided to the peripheral because: PERxEN = PERxLPEN = 1 and **lsi_ck** or **lse_ck** are selected and enabled. The bus interface clock is not provided as the CPU is in Stop.
1	1	hsi_ker_ck or csi_ker_ck	1		CK	-	Kernel clock is provided to the peripheral because: **req_ker_perx** = 1, and PERxEN = PERxLPEN = 1 and **hsi_ker_ck** or **csi_ker_ck** are selected and enabled. The bus interface clock is not provided as the CPU is in Stop.
1	1	hsi_ker_ck or csi_ker_ck	0		-	-	No clock provided to the peripheral because CPU is in Stop, and no kernel clock request pending

As a summary, we can state that the kernel clock is provided to the peripherals when the following conditions are met:

1. The CPU is in Run mode, and the peripheral is enabled.
2. The CPU is in Sleep mode and the peripheral is enabled with PERxLPEN = 1.
3. The CPU is in Stop mode, the peripheral is enabled with PERxLPEN = 1, the peripheral generates a kernel clock request and the selected clock is **hsi_ker_ck** or **csi_ker_ck**.
4. The CPU is in Stop mode, the peripheral is enabled with PERxLPEN = 1 and the kernel source clock of the peripheral is **lse_ck** or **lsi_ck**.

The bus interface clock is provided to the peripherals only when conditions 1 or 2 are met.

The RNG block is not clocked as other peripherals, the table hereafter shows the way RNG is clocked.

Table 63. RNG clock enabling

RNGEN	RNGLPEN	rng_ker_ckreq	CPU state	hsi48_ker_ck	rcc_rng_hclk3	Comments
0	X	X	X	-	-	No clock provided to the peripheral, because RNGEN=0.
1	X	0	Run	-	CK	**hsi48_ker_ck** not provided because the kernel clock request is not asserted. hclk3 clock is provided.
1	X	1		CK	CK	Both **hsi48_ker_ck** and hclk3 are provided because the kernel clock request is asserted, and RNGEN = 1.
1	0	X	Sleep	-	-	No clock provided, because the CPU is in Sleep and RNGLPEN = 0.
1	1	0		-	CK	**hsi48_ker_ck** not provided because the kernel clock request is not asserted. hclk3 clock is provided.
1	1	1		CK	CK	Both **hsi48_ker_ck** and hclk3 are provided because the kernel clock request is asserted, RNGEN = 1 and RNGLPEN = 1.
1	X	X	Stop	-	-	No clock provided because the CPU is in Stop mode.

7.5.13 CPU and bus matrix clock gating control

The clocks of the CPU, AHB and AXI bridges and APB buses are enabled according to the rules hereafter:

- The CPU clock (**rcc_cpu_ck**) is enabled when the CPU is in Run or Sleep mode.
- The AXI bridge clock is enabled when the CPU is in Run or Sleep mode.
- Each AXI master and slave has an independent clock gating activated by default, to further reduce the power consumption. When activated the clock is automatically enabled on bus transaction request. The *RCC AXI clocks gating disable register (RCC_CKGDISR)* allows the application to control this behavior.
- The AHB bus matrix clock is enabled if one of the following conditions is met:
  - The CPU is in Run mode.
  - The CPU is in Sleep mode, and at least one peripheral connected to this bridge have both its PERxEN and PERxLPEN set to 1.
  - The CPU is in Sleep mode, and either AHB1, AHB2, AHB3, AHB4 or AHB5 has its clock enabled.
- The clocks of the bridges AHB1, 2, 3, 4, 5 are enabled when one of the following conditions is met:
  - The CPU is in Run mode.
  - The CPU is in Sleep mode with at least one peripheral connected to this bus having both its PERxEN and PERxLPEN set to 1.
  - The CPU is in Sleep mode and the APB bus connected to the AHB bridge has its clock enabled.
- The APB1, 2, 4 clock buses are enabled when one of the following conditions is met:
  - The CPU is in Run mode.
  - When the CPU is in Sleep mode with at least one peripheral connected to this bus having both its PERxEN and PERxLPEN set to 1.

7.5.14 Low-power emulation modes

In order to ease the debugging of the circuit, the RCC is able to handle an emulation mode for Stop and Standby modes.

Sleep emulation mode

The Sleep emulation mode is controlled by the DBG_SLEEP bit of the DBGMCU_CR register. When the processor goes to Sleep with DBG_SLEEP = 1, then the processor clock, the clocks of all enabled peripherals, debug parts, and interconnect are maintained active.

Stop emulation mode

The Stop emulation mode is controlled by the DBG_STOP bit of the DBGMCU_CR register. When the processor goes to Stop with DBG_STOP = 1, then:

- The PWR and the RCC remain active.
- The V_{DDCORE} voltage is set to the minimum VOS value.
- The RCC selects HSI or CSI as system clock, depending on the STOPWUCK bit.
- The processor clock is gated (rcc_cpu_ck, see *Figure 48*).
- The free-running clock is generated (rcc_fclk, see *Figure 48*), and all the debug parts remain clocked.
- All peripheral clocks are gated. Peripherals can still receive a kernel clock if they request it, and if the kernel clock source is an RC oscillator. They also receive the kernel clock if the kernel clock source is LSI or LSE.
- The interconnect clocks are gated.

When a wakeup event occurs:

- The V_{DDCORE} voltage remains to the minimum VOS value.
- The peripheral that wakes up the system sends the interrupt to the NVIC via the EXTI.
- The CPU exits from STOP and the RCC provides the processor clock and interconnect clocks.

Standby emulation mode

The Standby emulation mode is controlled by the DBG_STANDBY bit of the DBGMCU_CR register.

When the system goes to Standby with DBG_STANDBY = 1, then:

- The V_{DDCORE} voltage is not switched-off, but remains to the minimum VOS value.
- All the V_{DDCORE} domain is under reset except the debug part.
- The debug part is clocked.
- Peripherals on V_{BAT} or V_{DD} domains are not reset.

When the system exits from Standby, then:

- Option bytes are reloaded.
- The reset of the V_{DDCORE} domain is released.
- The Standby flag is updated as well.

7.6 RCC interrupts

The RCC provides three interrupt lines:

- **rcc_it**: general interrupt line providing events when the PLLs are ready or when the oscillators are ready
- **rcc_hsecss_it**: interrupt line dedicated to the failure detection of the HSE CSS (clock security system)
- **rcc_lsecss_it**: interrupt line dedicated to the failure detection of the LSE CSS

The interrupt enable is controlled by *RCC clock source interrupt enable register (RCC_CIER)*, except for the HSE CSS failure. When the HSE CSS feature is enabled, it is not possible to mask the interrupt generation.

The interrupt flags can be checked via *RCC clock source interrupt flag register (RCC_CIFR)*, and these flags can be cleared via *RCC clock source interrupt clear register (RCC_CICR)*.

Note: The interrupt flags are not relevant if the corresponding interrupt enable bit is not set.

Table 64 gives a summary of the interrupt sources and the way to control them.

Table 64. Interrupt sources and control

Interrupt source	Description	Interrupt enable	Action to clear interrupt	Interrupt line
LSIRDYF	LSI ready	LSIRDYIE	Set LSIRDYC to 1	rcc_it
LSERDYF	LSE ready	LSERDYIE	Set LSERDYC to 1	
HSIDRYF	HSI ready	HSIDRYIE	Set HSIRDYC to 1	
HSERDYF	HSE ready	HSERDYIE	Set HSERDYC to 1	
CSIRDYF	CSI ready	CSIRDYIE	Set CSIRDYC to 1	
HSI48RDYF	HSI48 ready	HSI48RDYIE	Set HSI48RDYC to 1	
PLL1RDYF	PLL1 ready	PLL1RDYIE	Set PLL1RDYC to 1	
PLL2RDYF	PLL2 ready	PLL2RDYIE	Set PLL2RDYC to 1	
PLL3RDYF	PLL3 ready	PLL3RDYIE	Set PLL3RDYC to 1	
LSECSSF	LSE CSS failure	LSECSSFIE (1)	Set LSECSSC to 1	rcc_lsecss_it
HSECSSF	HSE CSS failure	- (2)	Set HSECSSC to 1	rcc_hsecss_it

1. The security system feature must also be enabled (LSECSSON = 1), in order to generate interrupts.
2. It is not possible to mask this interrupt when the security system feature is enabled (HSECSSON = 1).

7.7 RCC Programming examples

7.7.1 PLL programming procedure

PLL initialization procedure

The recommended initialization sequence of the PLLs for the integer and fractional mode is given hereafter:

- The PLL is supposed to be disabled: PLLxPEN, PLLxQEN, PLLxREN, PLLxSEN, PLLxTEN, PLLxON, and PLLxRDY set to 0, if it is not the case, refer to *PLL disabling procedure*.
- Enable the wanted clock source (HSE, HSI, or CSI) and wait for the ready flag.
- Select the clock source via PLLSCR into *RCC PLLs clock source selection register (RCC_PLLCKSELR)*.
- Initialize the pre-dividers (DIVMx) in order to provide a valid reference clock to the PLL. DIVMx are located into *RCC PLLs clock source selection register (RCC_PLLCKSELR)*.
- Configure the PLL:
  - In integer or clock spreading mode the application must ensure that a 0 is loaded into the SDM. This can be done as follows:
    - Set PLLxFRACLE to 0
    - Write FRACN to 0
    - Set PLLxFRACLE to 1
    - Write FRACN to 0
    - Wait at least 5 µs
  - In fractional mode, the application must load the SDM with the wanted value (FracInitValue) by executing a sequence similar to the integer mode, but with FRACN = FracInitValue.
  - Program the fields PLLxRGE, PLLxVCOSEL, SSCG_CTRL, DIVxN, DIVxP, DIVxQ, DIVxR, DIV2S and DIV2T.
  - If the clock spreading mode is used, also program the MODPER, INCSTEP, SPREADSEL, TPDFNDIS, RPDFNDIS fields, and set PLLxSSCGEN to 1.
  - If the clock spreading mode is not used, set PLLxSSCGEN to 0.
- Enable the PLL:
  - Set the corresponding PLLxON bit to 1, and wait for PLLxRDY bit set to 1,
  - If the PLL is in fractional mode, the FRACLE bit must not be set back to 0 as long as PLLxRDY = 0.
- When the PLLxRDY bit is equal to 1, the application can enable the wanted outputs by setting PLLxPEN, PLLxQEN, PLLxREN, PLLxSEN, PLLxTEN to 1.
- If the application intends to tune the PLL frequency on-the-fly (only possible in fractional mode), then:
  - Set PLLxFRACLE to 0,
    When FRACLE = 0, the sigma delta modulator is still operating with the current value,
  - Write the new value into FRACN[12:0] (FracValue(n)),
  - Set PLLxFRACLE to 1, in order to latch the content of FRACN[12:0] into the SDM.
  - Wait at least 5 µs.

Note: When the PLLxRDY goes to 1, it means that the PLLx output frequency is within 2% of its target value.

Reprogramming post-dividers

When the PLLs are enabled, it is possible to change the values of post-dividers (DIVP, DIVQ, DIVR, DIVS or DIVT) without disabling the corresponding PLLx, by performing the following sequence:

- Disable the wanted output by setting the bit PLLxPEN, PLLxQEN, PLLxREN, PLLxSEN or PLLxTEN to 0,
- Change the value of the corresponding post-divider (DIVP, DIVQ, DIVR, DIVS or DIVT),
- Re-enable this output by setting the bit PLLxPEN, PLLxQEN, PLLxREN, PLLxSEN or PLLxTEN to 1.

PLL disabling procedure

The recommended disabling sequence is given hereafter:

- Disable all the post-dividers by setting PLLxPEN, PLLxQEN, PLLxREN, PLLxSEN and PLLxTEN bits of PLLx to 0,
- Disable the PLL: set PLLxON bit of PLLx to 0,
- Wait for PLLxRDY = 0.

Using the SSCG block

For example, if an application needs to generate with the PLL2, a clock signal having the following characteristics:

- A frequency at DIVS output (**FpII2_s_ck**) of 200 MHz Max,
- With a modulation depth of 0.5% peak (M_D),
- With a frequency modulation of 28 kHz (F_{MOD}).

The assumption is that the crystal oscillator is 8 MHz (F_{hse_ck}).

DIVM2 is programmed to 0 in order to provide a reference clock of 8 MHz to the PLL2 (F_{ref2_ck}). The VCOH must be selected when SSCG is used.

From an 8 MHz reference clock, it is possible to generate 200 MHz, by programming the PLL2 as follows: (DIVN+1) = 50, (Fvco = 800 MHz), (DIVS+1) = 4.

The application has to compute the following parameters:

Computing MODPER:

$$MODPER = ROUND\left(\frac{F_{ref_ck}}{4 \times F_{MOD}}\right) = ROUND\left(\frac{8 \times 10^6}{4 \times 28 \times 10^3}\right) = 71$$

Computing INCSTEP:

$$INCSTEP = ROUND\left(\frac{(2^{15} - 1) \times M_D \times (DIVN + 1)}{100 \times 5 \times MODPER}\right) = ROUND\left(\frac{32767 \times 0.5 \times (49 + 1)}{100 \times 5 \times 71}\right) = 23$$

Check that MODPER x INCSTEP is lower than (2^{15}-1)

In the example, MODPER x INCSTEP = 1633 ==> OK

Due to rounding operations, the modulation period and the modulation depth do not completely match the target. The real modulation depth is:

$$M_D\,(\%) = \frac{MODPER \times INCSTEP \times 100 \times 5}{(2^{15} - 1) \times (DIVN + 1)} = \frac{71 \times 23 \times 100 \times 5}{(2^{15} - 1) \times (49 + 1)} = 0.498\ \%$$

And the modulation frequency (Fmod) is:

$$F_{mod} = \frac{F_{ck_ref}}{4 \times MODPER} = \frac{8 \times 10^6}{4 \times 80} = 28.2\ \text{kHz}$$

The down-spread modulation must be used (SPREADSEL = 1) to keep the maximum frequency to a value lower than 200 MHz.

7.7.2 Frequency configuration examples

This section gives various frequency settings for CPU and XSPI interfaces, combined with audio constraints. The case without audio constraints is not shown as it is obvious that PLL1 and PLL2 can provide the expected frequency. If audio applications are required, several options are offered to the user:

- Generate the kernel clock with PLL1, PLL2 or PLL3
- Use the external audio clock input (AUDIOCLK)
- Use oscillators

In this section, we explore the cases where PLL1 or PLL2 are used for audio.

Note as well that PLL2 is also used for XSPIs, if it is shared for audio application, it is not recommended to activate the SSCG.

The configurations listed in the tables hereafter are not exhaustive, many other settings are available. For example if the clock provided to the I2S must be shared with FDCAN, then the application would like to generate a clock around 100 MHz to insure good working conditions for FDCAN. These considerations are not taken into account in these tables.

Note also that in order to get the best clock quality at PLL output, F_{REF} must be as close as possible to 16 MHz. This is not always easy to do for PLLs in integer mode, but can be done easily when the PLLs are in fractional mode (as shown in the tables).

Table 65. Clock configuration examples with PLLs in integer mode

Configuration	HSE (MHz)	DIVM+1	F_{REF} (MHz) (1)	PLLxRGE	DIVN+1	FRACV	F_{VCO} (MHz) (2)	DIVP+1	DIVQ+1	DIVS/T+1	F_{CPU} (MHz)(3)	F_{XSPI} (MHz)(4)	F_{AUDIO} (MHz)(5)	F_{CPU} accuracy (%)	F_{XSPI} accuracy (%)	F_{AUDIO} accuracy (ppm)
PLL1 CPU: 500 MHz Audio: 48 kHz	16	7	2.286	1	215	0	982.9	1	4		491.4	-	122.86	-1.7	-	-186
	24	11	2.182	1	214	0	933.8	1	19		466.9	-	24.574	-6.6	-	-75
PLL1 CPU: 500 MHz Audio: 44.1 kHz	16	3	5.333	2	91	0	970.7	1	43		485.3	-	11.287	-3	-	-246
	24	10	2.400	1	207	0	993.6	1	4		496.8	-	124.2	-0.6	-	116
PLL1 CPU: 550 MHz Audio: 48 kHz	16	5	3.200	1	169	0	1081.6	1	11		540.8	-	49.164	-1.7	-	237
	24	2	12.000	3	43	0	1032.0	1	6		516.0	-	86.00	-6.2	-	-186
PLL1 CPU: 550 MHz Audio: 44.1 kHz	16	8	2.000	1	271	0	1084.0	1	6		542.0	-	90.333	-1.5	-	183
	24	12	2.000	1	271	0	1084.0	1	6		542.0	-	90.333	-1.5	-	183
PLL1 CPU: 400 MHz Audio: 48 kHz	16	7	2.286	1	172	0	786.2	1	4		393.1	-	98.290	-1.7	-	-186
	24	7	3.429	1	223	0	1597.7	2	13		399.4	-	61.451	-0.2	-	-172
PLL1 CPU: 400 MHz Audio: 44.1 kHz	24	11	2.182	1	357	0	1557.8	2	23		389.5	-	33.866	-2.6	-	-94

1. F_{REF} is the reference frequency at PLL1 or PLL2 inputs.
2. F_{VCO} is the clock frequency at VCO output of PLL1 or PLL2.
3. F_{CPU} is the clock frequency provided to the CPU (rcc_cpu_ck).
4. $F_{XSPI:}$ is the kernel clock frequency provided to the XSPI1 or XSPI2 (xspi_ker_ck).
5. F_{AUDIO} is the kernel clock frequency provided to the audio blocks such as ADF, SPI/I2Sx.
   For a 48 kHz stream, the expected kernel clock is 48 kHz x 256 x k, with k integer.
   For a 44.1 kHz stream, the expected kernel clock is 44.1 kHz x 256 x k, with k integer.

Table 66. Clock configuration examples with PLLs in fractional mode(1)

Configuration	HSE (MHz)	DIVM+1	F_{REF} (MHz) (2)	DIVN+1	FRACV	F_{VCO} (MHz) (3)	DIVP+1	DIVQ+1	DIVS/T+1	F_{CPU} (MHz)(4)	F_{XSPI} (MHz)(5)	F_{AUDIO} (MHz)(6)	F_{CPU} accuracy (%)	F_{XSPI} accuracy (%)	F_{AUDIO} accuracy (ppm)
PLL1 CPU: 500 MHz Audio: 48 kHz	16	1	16.0	30	5898	983.0	1	4	-	491.5	-	122.88	-1.7	-	-1
	24	2	12.0	40	7864										
PLL1 CPU: 500 MHz Audio: 44.1 kHz	16	1	16.0	31	380	993.5	1	4		496.7		124.2	-0.6		-0.4
	24	2	12.0	41	3237										-1.4
PLL1 CPU: 550 MHz Audio: 48 kHz	16	1	16.0	33	6488	1081.3	1	11	-	540.7	-	49.15	-1.7	-	0.2
	24	2	12.0	45	459										0.7
PLL1 CPU: 550 MHz Audio: 44.1 kHz	16	1	16.0	33	7117	1083.8	1	6		541.9		90.32	-1.5		-0.7
	24	2	12.0	45	1298										1.0
PLL1 CPU: 400 MHz Audio: 48 kHz	16	1	16.0	49	1245	1572.9	2	8	-	393.2	-	98.30	-1.7	-	0.5
	24	2	12.0	65	4391										0.1
PLL1 CPU: 400 MHz Audio: 44.1 kHz	16	1	16.0	49	3212	1580.5	2	70		395.1		112.90	1.2		-1.8
	24	2	12.0	65	7012										-0.7

1. Note that for all these configurations, PLLxRGE is always equal to 3.
2. F_{REF} is the reference frequency at PLL1 or PLL2 inputs.
3. F_{VCO} is the clock frequency at VCO output of PLL1 or PLL2.
4. F_{CPU} is the clock frequency provided to the CPU (rcc_cpu_ck).
5. F_{XSPI} is the kernel clock frequency provided to the XSPI1 or XSPI2 (xspi_ker_ck).
6. F_{AUDIO} is the kernel clock frequency provided to the SPI/I2Sx.
   For a 48 kHz stream, the expected kernel clock is 48 kHz x 256 x k, with k integer.
   For a 44.1 kHz stream, the expected kernel clock is 44.1 kHz x 256 x k, with k integer.

7.8 RCC registers

7.8.1 RCC source control register (RCC_CR)

Address offset: 0x000

Reset value: 0x0000 0025

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PLL3RDY	PLL3ON	PLL2RDY	PLL2ON	PLL1RDY	PLL1ON	Res.	Res.	Res.	HSECSSON	HSEEXT	HSEBYP	HSERDY	HSEON
		r	rw	r	rw	r	rw				rw	rw	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	HSI48RDY	HSI48ON	Res.	Res.	CSIKERON	CSIRDY	CSION	Res.	HSIDIVF	HSIDIV[1:0]		HSIRDY	HSIKERON	HSION
		r	rw			rw	r	rw		r	rw	rw	r	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **PLL3RDY**: PLL3 clock ready flag
Set by hardware to indicate that the PLL3 is locked.
0: PLL3 unlocked (default after reset)
1: PLL3 locked

Bit 28 **PLL3ON**: PLL3 enable
Set and cleared by software to enable PLL3.
Cleared by hardware when entering Stop or Standby mode.
0: PLL3 OFF (default after reset)
1: PLL3 ON

Bit 27 **PLL2RDY**: PLL2 clock ready flag
Set by hardware to indicate that the PLL2 is locked.
0: PLL2 unlocked (default after reset)
1: PLL2 locked

Bit 26 **PLL2ON**: PLL2 enable
Set and cleared by software to enable PLL2.
Cleared by hardware when entering Stop or Standby mode. Note that the hardware prevents writing this bit to 0, if FMCCKP = 1, or XSPICKP = 1.
0: PLL2 OFF (default after reset)
1: PLL2 ON

Bit 25 **PLL1RDY**: PLL1 clock ready flag
Set by hardware to indicate that the PLL1 is locked.
0: PLL1 unlocked (default after reset)
1: PLL1 locked

Bit 24 **PLL1ON**: PLL1 enable
Set and cleared by software to enable PLL1.
Cleared by hardware when entering Stop or Standby mode. Note that the hardware prevents writing this bit to 0, if the PLL1 output is used as the system clock (SW=3) or if FMCCKP = 1, or if XSPICKP = 1.
0: PLL1 OFF (default after reset)
1: PLL1 ON

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **HSECSSON**: HSE clock security system enable

Set by software to enable clock security system (CSS) on HSE.

Clearing this bit has no effect, the CSS is disabled by a system reset or when the system enters in Standby mode.

When HSECSSON is set, the clock detector is enabled by hardware when the HSE is ready and disabled by hardware if an oscillator failure is detected.

0: CSS on HSE OFF (clock detector OFF) (default after reset)

1: CSS on HSE ON (clock detector ON if the HSE oscillator is stable, OFF if not).

Bit 19 **HSEEXT**: external high speed clock type in Bypass mode

Set and reset by software to select the external clock type (analog or digital).

The external clock must be enabled with the HSEON bit to be used by the device.

The HSEEXT bit can be written only if the HSE oscillator is disabled.

0: HSE in analog mode (default after reset)

1: HSE in digital mode

Bit 18 **HSEBYP**: HSE clock bypass

Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit to be used by the device.

The HSEBYP bit can be written only if the HSE oscillator is disabled.

0: HSE oscillator not bypassed (default after reset)

1: HSE oscillator bypassed with an external clock

Bit 17 **HSERDY**: HSE clock ready flag

Set by hardware to indicate that the HSE oscillator is stable.

0: HSE clock is not ready (default after reset)

1: HSE clock is ready

Bit 16 **HSEON**: HSE clock enable

Set and cleared by software.

Cleared by hardware to stop the HSE when entering Stop or Standby mode.

This bit cannot be cleared if the HSE is used directly (via SW mux) as system clock, or if the HSE is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1) or if FMCCKP = 1, or if XSPICKP = 1.

0: HSE is OFF (default after reset)

1: HSE is ON

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

Bit 13 **HSI48RDY**: HSI48 clock ready flag

Set by hardware to indicate that the HSI48 oscillator is stable.

0: HSI48 clock is not ready (default after reset)

1: HSI48 clock is ready

Bit 12 **HSI48ON**: HSI48 clock enable

Set by software and cleared by software or by the hardware when the system enters to Stop or Standby mode.

0: HSI48 is OFF (default after reset)

1: HSI48 is ON

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **CSIKERON**: CSI clock enable in Stop mode

Set and reset by software to force the CSI to ON, even in Stop mode, in order to be quickly available as kernel clock for some peripherals. This bit has no effect on the value of CSION.

0: No effect on CSI (default after reset)

1: CSI is forced to ON even in Stop mode

Bit 8 **CSIRDY**: CSI clock ready flag

Set by hardware to indicate that the CSI oscillator is stable. This bit is activated only if the RC is enabled by CSION (it is not activated if the CSI is enabled by CSIKERON or by a peripheral request).

0: CSI clock is not ready (default after reset)

1: CSI clock is ready

Bit 7 **CSION**: CSI clock enable

Set and reset by software to enable/disable CSI clock for system and/or peripheral.

Set by hardware to force the CSI to ON when the system leaves Stop mode, if STOPWUCK = 1 or STOPKERWUCK = 1.

This bit cannot be cleared if the CSI is used directly (via SW mux) as system clock, or if the CSI is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1) or if FMCCKP = 1, or if XSPICKP = 1.

0: CSI is OFF (default after reset)

1: CSI is ON

Bit 6 Reserved, must be kept at reset value.

Bit 5 **HSIDIVF**: HSI divider flag

Set and reset by hardware.

As a write operation to HSIDIV has not an immediate effect on the frequency, this flag indicates the current status of the HSI divider. HSIDIVF goes immediately to 0 when HSIDIV value is changed, and is set back to 1 when the output frequency matches the value programmed into HSIDIV.

0: New division ratio not yet propagated to **hsi(_ker)_ck** (default after reset)

1: **hsi(_ker)_ck** clock frequency reflects the new HSIDIV value (default register value when the clock setting is completed)

Bits 4:3 **HSIDIV[1:0]**: HSI clock divider

Set and reset by software.

These bits allow selecting a division ratio in order to configure the wanted HSI clock frequency. The HSIDIV cannot be changed if the HSI is selected as reference clock for at least one enabled PLL (PLLxON bit set to 1). In that case, the new HSIDIV value is ignored.

00: Division by 1, **hsi(_ker)_ck** = 64 MHz (default after reset)

01: Division by 2, **hsi(_ker)_ck** = 32 MHz

10: Division by 4, **hsi(_ker)_ck** = 16 MHz

11: Division by 8, **hsi(_ker)_ck** = 8 MHz

Bit 2 **HSIRDY**: HSI clock ready flag

Set by hardware to indicate that the HSI oscillator is stable.

0: HSI clock is not ready (default after reset)

1: HSI clock is ready

Bit 1 **HSIKERON**: HSI clock enable in Stop mode

Set and reset by software to force the HSI to ON, even in Stop mode, in order to be quickly available as kernel clock for peripherals. This bit has no effect on the value of HSION.

0: No effect on HSI (default after reset)

1: HSI is forced to ON even in Stop mode

Bit 0 **HSION**: HSI clock enable

Set and cleared by software.

Set by hardware to force the HSI to ON when the product leaves Stop mode, if STOPWUCK = 0 or STOPKERWUCK = 0.

Set by hardware to force the HSI to ON when the product leaves Standby mode or in case of a failure of the HSE which is used as the system clock source.

This bit cannot be cleared if the HSI is used directly (via SW switch) as system clock, or if the HSI is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1) or if FMCCKP = 1, or if XSPICKP = 1.

0: HSI is OFF

1: HSI is ON (default after reset)

7.8.2 RCC clock protection register (RCC_CKPROTR)

Address offset: 0x100

Reset value: 0x0000 0000

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

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:12 **FMCSWP[2:0]**: FMC kernel clock switch position

Set by hardware.

This field can be used to verify the real position of FMC kernel switch selector.

000: The switch is in neutral mode and output clock is gated (default after reset)

001: The switch is selecting hclk5

010: The switch is selecting pll1_q_ck

011: The switch is selecting pll2_r_ck

100: The switch is selecting hsi_ker_ck

101: The switch is in recovery position (hclk5/4)

Bits 11:7 Reserved, must be kept at reset value.

Bits 6:4 **XSPISWP[2:0]**: XSPI kernel clock switch position
Set by hardware.
This field can be used to verify the real position of XSPI kernel switch selector.
000: The switch is in neutral mode and output clock is gated (default after reset)
001: The switch is selecting hclk5
010: The switch is selecting pll2_s_ck
011: The switch is selecting pll2_t_ck
100: The switch is in recovery position (hclk5/4)

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **FMCCKP**: FMC clock protection
Set and cleared by software.
When set to 1, this bit prevents disabling accidentally the FMC. The following fields cannot be modified when this bit is set to 1:
PLL1ON, PLL2ON, PLL1QEN, PLL2REN, HSEON, HSION, CSION, FMCEN, FMCLPEN, FMCRST.
0: Clock protection is disabled (default after reset)
1: Clock protection is enabled

Bit 0 **XSPICKP**: XSPI clock protection
Set and cleared by software.
When set to 1, this bit prevents disabling accidentally the XSPIs. The following fields cannot be modified when this bit is set to 1:
PLL2ON, PLL2SEN, PLL2TEN, HSEON, HSION, CSION, XSPIEN, XSPILPEN, XSPIRST.
0: Clock protection is disabled (default after reset)
1: Clock protection is enabled

7.8.3 RCC HSI calibration register (RCC_HSICFGR)

Address offset: 0x004

Reset value: 0x4000 0XXX

The reset value depends on the flash option bytes setting.

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

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **HSITRIM[6:0]**: HSI clock trimming
Set by software to adjust calibration.
HSITRIM field is added to the engineering option bytes loaded during reset phase (FLASH_HSI_opt) in order to form the calibration trimming value.
HSICAL = HSITRIM + FLASH_HSI_opt.
Note: The reset value of the field is 0x40.

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:0 **HSICAL[11:0]**: HSI clock calibration
Set by hardware by option byte loading.
Adjusted by software through trimming bits HSITRIM.
This field represents the sum of engineering option byte calibration value and HSITRIM bits value.

7.8.4 RCC clock recovery RC register (RCC_CRRCR)

Address offset: 0x008

Reset value: 0x0000 0XXX

The reset value depends on the flash option bytes setting.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	HSI48CAL[9:0]									
						r	r	r	r	r	r	r	r	r	r

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

Bits 9:0 **HSI48CAL[9:0]**: Internal RC 48 MHz clock calibration
Set by hardware by option byte loading.
Read-only.

7.8.5 RCC CSI calibration register (RCC_CSICFGR)

Address offset: 0x00C

Reset value: 0x2000 00XX

The reset value depends on the flash option bytes setting.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CSITRIM[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rw	rw	rw	rw	rw	rw								

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

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **CSITRIM[5:0]**: CSI clock trimming
Set by software to adjust calibration.
CSITRIM field is added to the engineering option bytes loaded during reset phase (FLASH_CSI_opt) in order to form the calibration trimming value.
CSICAL = CSITRIM + FLASH_CSI_opt.
Note: The reset value of the field is 0x20.

Bits 23:8 Reserved, must be kept at reset value.

Bits 7:0 **CSICAL[7:0]**: CSI clock calibration
Set by hardware by option byte loading.
Adjusted by software through trimming bits CSITRIM.
This field represents the sum of engineering option byte calibration value and CSITRIM bits value.

7.8.6 RCC clock configuration register (RCC_CFGR)

Address offset: 0x010

Reset value: 0x0000 0000

Access: Word, half-word only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2SEL[2:0]			MCO2PRE[3:0]				MCO1SEL[2:0]			MCO1PRE[3:0]				Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RTCPRE[5:0]						STOPKERWUCK	STOPWUCK	Res.	SWS[1:0]		Res.	SW[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw		r	r		rw	rw

Bits 31:29 **MCO2SEL[2:0]**: microcontroller clock output 2
Set and cleared by software. Clock source selection may generate glitches on MCO2.
It is highly recommended to configure these bits only after reset, before enabling the external oscillators and the PLLs.
000: System clock selected (**sys_ck**) (default after reset)
001: PLL2 oscillator clock selected (**pll2_p_ck**)
010: HSE clock selected (**hse_ck**)
011: PLL1 clock selected (**pll1_p_ck**)
100: CSI clock selected (**csi_ck**)
101: LSI clock selected (**lsi_ck**)
Others: Reserved

Bits 28:25 **MCO2PRE[3:0]**: MCO2 prescaler

Set and cleared by software to configure the prescaler of the MCO2. Modification of this prescaler may generate glitches on MCO2. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.

0000: Prescaler disabled (default after reset)
0001: Division by 1 (bypass)
0010: Division by 2
0011: Division by 3
0100: Division by 4
...
1111: Division by 15

Bits 24:22 **MCO1SEL[2:0]**: Microcontroller clock output 1

Set and cleared by software. Clock source selection may generate glitches on MCO1.
It is highly recommended to configure these bits only after reset, before enabling the external oscillators and the PLLs.

000: HSI clock selected (**hsi_ck**) (default after reset)
001: LSE oscillator clock selected (**lse_ck**)
010: HSE clock selected (**hse_ck**)
011: PLL1 clock selected (**pll1_q_ck**)
100: HSI48 clock selected (**hsi48_ck**)
Others: Reserved

Bits 21:18 **MCO1PRE[3:0]**: MCO1 prescaler

Set and cleared by software to configure the prescaler of the MCO1. Modification of this prescaler may generate glitches on MCO1. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.

0000: Prescaler disabled (default after reset)
0001: Division by 1 (bypass)
0010: Division by 2
0011: Division by 3
0100: Division by 4
...
1111: Division by 15

Bits 17:14 Reserved, must be kept at reset value.

Bits 13:8 **RTCPRE[5:0]**: HSE division factor for RTC clock

Set and cleared by software to divide the HSE to generate a clock for RTC.
Caution: The software must set these bits correctly to ensure that the clock supplied to the RTC is lower than 1 MHz. These bits must be configured if needed before selecting the RTC clock source.

000000: No clock (default after reset)
000001: No clock
000010: HSE/2
000011: HSE/3
000100: HSE/4
...
111110: HSE/62
111111: HSE/63

Bit 7 **STOPKERWUCK**: kernel clock selection after a wake up from system Stop

Set and reset by software to select the kernel wakeup clock from system Stop.

0: HSI selected as wake up clock from system Stop (default after reset)
1: CSI selected as wake up clock from system Stop

See *Section 7.5.8: Clock generation in Stop and Standby modes* for details.

Bit 6 **STOPWUCK**: system clock selection after a wake up from system Stop
Set and reset by software to select the system wakeup clock from system Stop.
The selected clock is also used as emergency clock for the clock security system (CSS) on HSE.
0: HSI selected as wake up clock from system Stop (default after reset)
1: CSI selected as wake up clock from system Stop
See *Section 7.5.8: Clock generation in Stop and Standby modes* for details.

Caution: STOPWUCK must not be modified when CSS is enabled (by HSECSSON bit) and the system clock is HSE (SWS = 10) or a switch on HSE is requested (SW =10).

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **SWS[1:0]**: system clock switch status
Set and reset by hardware to indicate which clock source is used as system clock.
00: HSI used as system clock (**hsi_ck**) (default after reset)
01: CSI used as system clock (**csi_ck**)
10: HSE used as system clock (**hse_ck**)
11: PLL1 used as system clock (**pll1_p_ck**)

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SW[1:0]**: system clock switch
Set and reset by software to select system clock source (**sys_ck**).
Set by hardware in order to force the selection of the HSI or CSI (depending on STOPWUCK selection) when leaving a system Stop mode or in case of failure of the HSE when used directly or indirectly as system clock.
00: HSI selected as system clock (**hsi_ck**) (default after reset)
01: CSI selected as system clock (**csi_ck**)
10: HSE selected as system clock (**hse_ck**)
11: PLL1 selected as system clock (**pll1_p_ck**)

7.8.7 RCC CPU domain clock configuration register (RCC_CDCFGR)

Address offset: 0x018

Reset value: 0x0000 0000

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

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

Bits 3:0 **CPRE[3:0]**: CPU domain core prescaler

Set and reset by software to control the CPU clock division factor.
Changing this division ratio has an impact on the frequency of the CPU clock and all bus matrix clocks.
After changing this prescaler value, it takes up to 16 periods of the slowest APB clock before the new division ratio is taken into account. The application can check if the new division factor is taken into account by reading back this register.
0000 - 0111: **sys_ck** not divided (default after reset)
1000: **sys_ck** divided by 2
1001: **sys_ck** divided by 4
1010: **sys_ck** divided by 8
1011: **sys_ck** divided by 16
1100: **sys_ck** divided by 64
1101: **sys_ck** divided by 128
1110: **sys_ck** divided by 256
1111: **sys_ck** divided by 512

Caution: Care must be taken when using the voltage scaling. Due to the propagation delay of the new division factor, after a prescaler factor change and before lowering the V_{CORE} voltage, this register must be read in order to check that the new prescaler value has been taken into account.

Depending on the clock source frequency and the voltage range, the software application must program a correct value in BMPRE to make sure that the system frequency does not exceed the maximum frequency.

7.8.8 RCC AHB clock configuration register (RCC_BMCFGR)

Address offset: 0x01C

Reset value: 0x0000 0000

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

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

Bits 3:0 **BMPRE[3:0]**: Bus matrix clock prescaler

Set and reset by software to control the division factor of **rcc_hclk[5:1]** and **rcc_aclk**. This group of clocks is also named **sys_bus_ck**. Changing this division ratio has an impact on the frequency of all bus matrix clocks.
0000 - 0111: **sys_bus_ck= sys_cpu_ck** (default after reset)
1000 - 1111: **sys_bus_ck** = **sys_cpu_ck** / 2

*Note: The new prescaler factor takes effect from 1 to 16 periods of the slowest APB clock among **rcc_pclk1,2,4** following an update of BMPRE.*

*Note: Note also that frequency of **rcc_hclk[5:1]** = **rcc_aclk = sys_bus_ck**.*

7.8.9 RCC APB clocks configuration register (RCC_APBCFGR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	PPRE4[2:0]			Res.	PPRE2[2:0]			Res.	PPRE1[2:0]		
					rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **PPRE4[2:0]**: CPU domain APB4 prescaler

Set and reset by software to control the division factor of **rcc_pclk4**.
The clock is divided by the new prescaler factor from 1 to 16 cycles of **sys_bus_ck** after PPRE4 write.
000 - 011: **rcc_pclk4** = **sys_bus_ck** (default after reset)
100: **rcc_pclk4** = **sys_bus_ck** / 2
101: **rcc_pclk4** = **sys_bus_ck** / 4
110: **rcc_pclk4** = **sys_bus_ck** / 8
111: **rcc_pclk4** = **sys_bus_ck** / 16

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **PPRE2[2:0]**: CPU domain APB2 prescaler

Set and reset by software to control the division factor of **rcc_pclk2**.
The clock is divided by the new prescaler factor from 1 to 16 cycles of **sys_bus_ck** after PPRE2 write.
000 - 011: **rcc_pclk2** = **sys_bus_ck** (default after reset)
100: **rcc_pclk2** = **sys_bus_ck** / 2
101: **rcc_pclk2** = **sys_bus_ck** / 4
110: **rcc_pclk2** = **sys_bus_ck** / 8
111: **rcc_pclk2** = **sys_bus_ck** / 16

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **PPRE1[2:0]**: CPU domain APB1 prescaler

Set and reset by software to control the division factor of **rcc_pclk1**.
The clock is divided by the new prescaler factor from 1 to 16 cycles of **sys_bus_ck** after PPRE1 write.
000 - 011: **rcc_pclk1** = **sys_bus_ck** (default after reset)
100: **rcc_pclk1** = **sys_bus_ck** / 2
101: **rcc_pclk1** = **sys_bus_ck** / 4
110: **rcc_pclk1** = **sys_bus_ck** / 8
111: **rcc_pclk1** = **sys_bus_ck** / 16

7.8.10 RCC PLLs clock source selection register (RCC_PLLCKSELR)

Address offset: 0x028

Reset value: 0x0202 0200

Access: Word only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	DIVM3[5:0]						Res.	Res.	DIVM2[5:4]	
						rw	rw	rw	rw	rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIVM2[3:0]				Res.	Res.	DIVM1[5:0]						Res.	Res.	PLLSRC[1:0]	
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw			rw	rw

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

Bits 25:20 **DIVM3[5:0]:** prescaler for PLL3

Set and cleared by software to configure the prescaler of the PLL3.
The hardware does not allow any modification of this prescaler when PLL3 is enabled (PLL3ON = 1).
In order to save power when PLL3 is not used, the value of DIVM3 must be set to 0.
000000: Prescaler disabled
000001: Division by 1 (bypass)
000010: Division by 2
000011: Division by 3
...
100000: Division by 32 (default after reset)
...
111111: Division by 63

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:12 **DIVM2[5:0]**: prescaler for PLL2

Set and cleared by software to configure the prescaler of the PLL2.
The hardware does not allow any modification of this prescaler when PLL2 is enabled (PLL2ON = 1).
In order to save power when PLL2 is not used, the value of DIVM2 must be set to 0.
000000: Prescaler disabled
000001: Division by 1 (bypass)
000010: Division by 2
000011: Division by 3
...
100000: Division by 32 (default after reset)
...
111111: Division by 63

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:4 **DIVM1[5:0]**: prescaler for PLL1

Set and cleared by software to configure the prescaler of the PLL1.
The hardware does not allow any modification of this prescaler when PLL1 is enabled (PLL1ON = 1).
In order to save power when PLL1 is not used, the value of DIVM1 must be set to 0.
000000: Prescaler disabled
000001: Division by 1 (bypass)
000010: Division by 2
000011: Division by 3
...
100000: Division by 32 (default after reset)
...
111111: Division by 63

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **PLLSRC[1:0]:** DIVMx and PLLs clock source selection

Set and reset by software to select the PLL clock source.
These bits can be written only when all PLLs are disabled.
In order to save power, when no PLL is used, PLLSRC must be set to 11.
00: HSI selected as PLL clock (**hsi_ck**) (default after reset)
01: CSI selected as PLL clock (**csi_ck**)
10: HSE selected as PLL clock (**hse_ck**)
11: No clock send to DIVMx divider and PLLs

7.8.11 RCC PLLs configuration register (RCC_PLLCFGR)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PLL3SEN	PLL3REN	PLL3QEN	PLL3PEN	PLL3RGE[1:0]		PLL3SSCGEN	PLL3VCOSEL	PLL3FRACEN	Res.	PLL2TEN	PLL2SEN	PLL2REN	PLL2QEN	PLL2PEN
	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL2RGE[1:0]		PLL2SSCGEN	PLL2VCOSEL	PLL2FRACLEN	Res.	Res.	PLL1SEN	Res.	PLL1QEN	PLL1PEN	PLL1RGE[1:0]		PLL1SSCGEN	PLL1VCOSEL	PLL1FRACEN
rw	rw	rw	rw	rw			rw		rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **PLL3SEN**: PLL3 DIVS divider output enable

Set and reset by software to enable the **pll3_s_ck** output of the PLL3.
To save power, PLL3SEN must be set to 0 when the **pll3_s_ck** is not used.
0: **pll3_s_ck** output disabled (default after reset)
1: **pll3_s_ck** output enabled

Bit 29 **PLL3REN**: PLL3 DIVR divider output enable

Set and reset by software to enable the **pll3_r_ck** output of the PLL3.
To save power, PLL3REN and DIVR bits must be set to 0 when the **pll3_r_ck** is not used.
0: **pll3_r_ck** output disabled (default after reset)
1: **pll3_r_ck** output enabled

Bit 28 **PLL3QEN**: PLL3 DIVQ divider output enable
Set and reset by software to enable the **pll3_q_ck** output of the PLL3.
To save power, PLL3REN and DIVR bits must be set to 0 when the **pll3_q_ck** is not used.
0: **pll3_q_ck** output disabled (default after reset)
1: **pll3_q_ck** output enabled

Bit 27 **PLL3PEN**: PLL3 DIVP divider output enable
Set and reset by software to enable the **pll3_p_ck** output of the PLL3.
To save power, PLL3REN and DIVR bits must be set to 0 when the **pll3_p_ck** is not used.
0: **pll3_p_ck** output disabled (default after reset)
1: **pll3_p_ck** output enabled

Bits 26:25 **PLL3RGE[1:0]**: PLL3 input frequency range
Set and reset by software to select the proper reference frequency range used for PLL3.
These bits can only be modified if PLL3ON = 0 in *RCC source control register (RCC_CR)*.
00: PLL3 input (**ref3_ck**) clock range frequency between 1 and 2 MHz (default after reset)
01: PLL3 input (**ref3_ck**) clock range frequency between 2 and 4 MHz
10: PLL3 input (**ref3_ck**) clock range frequency between 4 and 8 MHz
11: PLL3 input (**ref3_ck**) clock range frequency between 8 and 16 MHz

Bit 24 **PLL3SSCGEN**: PLL3 SSCG enable
Set and reset by software to enable the spread spectrum clock generator of PLL3, in order to reduce the number of EMI peaks.
0: SSCG disabled (default after reset)
1: SSCG enabled

Bit 23 **PLL3VCOSEL**: PLL3 VCO selection
Set and reset by software to select the proper VCO frequency range used for PLL3.
This bit can only be modified if PLL3ON = 0 in *RCC source control register (RCC_CR)*. It allows the application to select the VCO range:
– VCOH: working from 384 to 1672 MHz (F_{ref2_ck} must be between 2 and 16 MHz)
– VCOL: working from 150 to 420 MHz (F_{ref2_ck} must be between 1 and 2 MHz)
0: VCOH selected (default after reset)
1: VCOL selected

Bit 22 **PLL3FRACEN**: PLL3 fractional latch enable
Set and reset by software to latch the content of FRACN into the sigma-delta modulator.
In order to latch the FRACN value into the sigma-delta modulator, PLL3FRACEN must be set to 0, then set to 1. The transition 0 to 1 transfers the content of FRACN into the modulator.
Refer to *PLL initialization procedure on page 385* for additional information.
PLL3FRACEN must remain at 0 for at least 5 µs.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **PLL2TEN**: PLL2 DIVT divider output enable
Set and reset by software to enable the **pll2_t_ck** output of the PLL2.
To save power, PLL2TEN must be set to 0 when the **pll2_t_ck** is not used.
The hardware prevents writing this bit if XSPICKP = 1.
0: **pll2_t_ck** output disabled (default after reset)
1: **pll2_t_ck** output enabled

Bit 19 **PLL2SEN**: PLL2 DIVS divider output enable
Set and reset by software to enable the **pll2_s_ck** output of the PLL2.
To save power, PLL2SEN must be set to 0 when the **pll2_s_ck** is not used.
The hardware prevents writing this bit if XSPICKP = 1.
0: **pll2_s_ck** output disabled (default after reset)
1: **pll2_s_ck** output enabled

Bit 18 **PLL2REN**: PLL2 DIVR divider output enable
Set and reset by software to enable the **pll2_r_ck** output of the PLL2.
The hardware prevents writing this bit if FMCCKP = 1.
To save power, PLL3REN and DIVR bits must be set to 0 when the **pll3_r_ck** is not used.
0: **pll2_r_ck** output disabled (default after reset)
1: **pll2_r_ck** output enabled

Bit 17 **PLL2QEN**: PLL2 DIVQ divider output enable
Set and reset by software to enable the **pll2_q_ck** output of the PLL2.
To save power, PLL3QEN and DIVQ bits must be set to 0 when the **pll2_q_ck** is not used.
0: **pll2_q_ck** output disabled (default after reset)
1: **pll2_q_ck** output enabled

Bit 16 **PLL2PEN**: PLL2 DIVP divider output enable
Set and reset by software to enable the **pll2_p_ck** output of the PLL2.
To save power, PLL2PEN and DIVP bits must be set to 0 when the **pll2_p_ck** is not used.
0: **pll2_p_ck** output disabled (default after reset)
1: **pll2_p_ck** output enabled

Bits 15:14 **PLL2RGE[1:0]**: PLL2 input frequency range
Set and reset by software to select the proper reference frequency range used for PLL2.
These bits can only be modified if PLL2ON = 0 in *RCC source control register (RCC_CR)*.
00: PLL3 input (**ref2_ck**) clock range frequency between 1 and 2 MHz (default after reset)
01: PLL3 input (**ref2_ck**) clock range frequency between 2 and 4 MHz
10: PLL3 input (**ref2_ck**) clock range frequency between 4 and 8 MHz
11: PLL3 input (**ref2_ck**) clock range frequency between 8 and 16 MHz

Bit 13 **PLL2SSCGEN**: PLL2 SSCG enable
Set and reset by software to enable the Spread Spectrum Clock Generator of PLL2, in order to reduce the amount of EMI peaks.
0: SSCG disabled (default after reset)
1: SSCG enabled

Bit 12 **PLL2VCOSEL**: PLL2 VCO selection
Set and reset by software to select the proper VCO frequency range used for PLL2.
This bit can only be modified if PLL2ON = 0 in *RCC source control register (RCC_CR)*. It allows the application to select the VCO range:
– VCOH: working from 384 to 1672 MHz (F_{ref2_ck} must be between 2 and 16 MHz)
– VCOL: working from 150 to 420 MHz (F_{ref2_ck} must be between 1 and 2 MHz)
0: VCOH selected (default after reset)
1: VCOL selected

Bit 11 **PLL2FRACLEN**: PLL2 fractional latch enable
Set and reset by software to latch the content of FRACN into the sigma-delta modulator.
In order to latch the FRACN value into the sigma-delta modulator, PLL2FRACLEN must be set to 0, then set to 1. The transition 0 to 1 transfers the content of FRACN into the modulator.
Refer to *PLL initialization procedure on page 385* for additional information.
PLL2FRACLEN must remain at 0 for at least 5 µs.

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **PLL1SEN**: PLL1 DIVS divider output enable

Set and reset by software to enable the **pll1_s_ck** output of the PLL1.
To save power, PLL1SEN must be set to 0 when the **pll1_s_ck** is not used.
0: **pll1_s_ck** output disabled (default after reset)
1: **pll1_s_ck** output enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **PLL1QEN**: PLL1 DIVQ divider output enable

Set and reset by software to enable the **pll1_q_ck** output of the PLL1.
The hardware prevents writing this bit if FMCCKP = 1.
In order to save power, when the **pll1_q_ck** output of the PLL1 is not used, the **pll1_q_ck** must be disabled.
0: **pll1_q_ck** output disabled (default after reset)
1: **pll1_q_ck** output enabled

Bit 5 **PLL1PEN**: PLL1 DIVP divider output enable

Set and reset by software to enable the **pll1_p_ck** output of the PLL1.
The hardware prevents modifying this bit if the PLL1 output is used as the system clock (SW=3).
In order to save power, when the **pll1_p_ck** output of the PLL1 is not used, the **pll1_p_ck** must be disabled.
0: **pll1_p_ck** output disabled (default after reset)
1: **pll1_p_ck** output enabled

Bits 4:3 **PLL1RGE[1:0]**: PLL1 input frequency range

Set and reset by software to select the reference frequency range used for PLL1.
These bits can only be modified if PLL1ON = 0 in *RCC source control register (RCC_CR)*.
00: PLL1 input (**ref1_ck**) clock range frequency between 1 and 2 MHz (default after reset)
01: PLL1 input (**ref1_ck**) clock range frequency between 2 and 4 MHz
10: PLL1 input (**ref1_ck**) clock range frequency between 4 and 8 MHz
11: PLL1 input (**ref1_ck**) clock range frequency between 8 and 16 MHz

Bit 2 **PLL1SSCGEN**: PLL1 SSCG enable

Set and reset by software to enable the Spread Spectrum Clock Generator of PLL1, in order to reduce the amount of EMI peaks.
0: SSCG disabled (default after reset)
1: SSCG enabled

Bit 1 **PLL1VCOSEL**: PLL1 VCO selection

Set and reset by software to select the VCO frequency range used for PLL1.
This bit can only be modified if PLL1ON = 0 in *RCC source control register (RCC_CR)*. It allows the application to select the VCO range:

- VCOH: working from 384 to 1672 MHz (F_{ref1_ck} must be between 2 and 16 MHz)
- VCOL: working from 150 to 420 MHz (F_{ref1_ck} must be between 1 and 2 MHz)

0: VCOH selected (default after reset)
1: VCOL selected

Bit 0 **PLL1FRACEN**: PLL1 fractional latch enable

Set and reset by software to latch the content of FRACN into the sigma-delta modulator.
In order to latch the FRACN value into the sigma-delta modulator, PLL1FRACEN must be set to 0, then set to 1. The transition 0 to 1 transfers the content of FRACN into the modulator.
Refer to *PLL initialization procedure on page 385* for additional information.
PLL1FRACEN must remain at 0 for at least 5 µs.

7.8.12 RCC PLL1 dividers configuration register 1 (RCC_PLL1DIVR1)

Address offset: 0x030

Reset value: 0x0101 0280

Access: Word and half-word only

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

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

Bits 22:16 **DIVQ[6:0]**: PLL1 DIVQ division factor

Set and reset by software to control the frequency of the **pll1_q_ck** clock.

0000000: **pll1_q_ck** = **vco1_ck**

0000001: **pll1_q_ck** = **vco1_ck** / 2 (default after reset)

0000010: **pll1_q_ck** = **vco1_ck** / 3

0000011: **pll1_q_ck** = **vco1_ck** / 4

...

1111111: **pll1_q_ck** = **vco1_ck** / 128

Bits 15:9 **DIVP[6:0]**: PLL1 DIVP division factor

Set and reset by software to control the frequency of the **pll1_p_ck** clock.
0000000: **pll1_p_ck** = **vco1_ck**
0000001: **pll1_p_ck** = **vco1_ck** / 2 (default after reset)
0000010: Not allowed
0000011: **pll1_p_ck** = **vco1_ck** / 4
...
1111111: **pll1_p_ck** = **vco1_ck** / 128

Bits 8:0 **DIVN[8:0]**: multiplication factor for PLL1 VCO

Set and reset by software to control the multiplication factor of the VCO.
These bits can be written only when the PLL is disabled (PLL1ON = 0 in *RCC source control register (RCC_CR)*).
.....: Not used
0x006: Wrong configuration
0x007: DIVN = 8
...
0x080: DIVN = 129 (default after reset)
...
0x1A3: DIVN = 420
Others: Wrong configurations

Caution: The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is:

– 128 to 544 MHz if PLL1VCOSEL = 0
– 150 to 420 MHz if PLL1VCOSEL = 1

VCO output frequency = F_{ref1_ck} x DIVN, when fractional value 0 has been loaded into FRACN, with:

– DIVN between 8 and 420
– The input frequency F_{ref1_ck} must be between 1 and 16 MHz.

7.8.13 RCC PLL1 dividers configuration register 2 (RCC_PLL1DIVR2)

Address offset: 0x0C0

Reset value: 0x0000 0101

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

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

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

Bits 2:0 **DIVS[2:0]**: PLL1 DIVS division factor

Set and reset by software to control the frequency of the **pll1_s_ck** clock.
This post-divider performs divisions with 50% duty-cycle.
The duty-cycle of 50% is guaranteed only in the following conditions:

– With VCOL, if (DIVS+1) is even,
– With VCOH, for all DIVS values

These bits can be written only when the PLL1SEN = 0.
000: **pll1_s_ck** = **vco1_ck**
001: **pll1_s_ck** = **vco1_ck** / 2 (default after reset)
010: **pll1_s_ck** = **vco1_ck** / 3
011: **pll1_s_ck** = **vco1_ck** / 4
100: **pll1_s_ck** = **vco1_ck** / 5
101: **pll1_s_ck** = **vco1_ck** / 6
110: **pll1_s_ck** = **vco1_ck** / 7
111: **pll1_s_ck** = **vco1_ck** / 8

7.8.14 RCC PLL1 fractional divider register (RCC_PLL1FRACR)

Address offset: 0x034

Reset value: 0x0000 0000

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:3 **FRACN[12:0]**: fractional part of the multiplication factor for PLL1 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO.
These bits can be written at any time, allowing dynamic fine-tuning of the PLL1 VCO.

Caution: The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is:

– 128 to 544 MHz if PLL1VCOSEL = 0
– 150 to 420 MHz if PLL1VCOSEL = 1

VCO output frequency = F_{ref1_ck} x $\left(\text{DIVN} + (\text{FRACN} / 2^{13})\right)$, with

– DIVN between 8 and 420
– FRACN can be between 0 and 2^{13}- 1
– The input frequency F_{ref1_ck} must be between 1 and 16 MHz.

To change the FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:

– Set the bit PLL1FRACEN to 0.
– Write the new fractional value into FRACN.
– Set the bit PLL1FRACEN to 1
– Wait at least 5 µs.

Bits 2:0 Reserved, must be kept at reset value.

7.8.15 RCC PLL1 Spread Spectrum Clock Generator register (RCC_PLL1SSCGR)

Address offset: 0x0CC

Reset value: 0x0000 0000

Access: Word and half-word only

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

Bit 31 Reserved, must be kept at reset value.

Bits 30:16 **INCSTEP[14:0]**: Modulation Depth Adjustment for PLL1

Set and reset by software to adjust the modulation depth of the clock spreading generator.

Bit 15 **SPREADSEL**: Spread spectrum clock generator mode for PLL1

Set and reset by software to select the clock spreading mode.

0: Center-spread modulation selected (default after reset)

1: Down-spread modulation selected

Bit 14 **RPDFNDIS**: Dithering RPDF noise control for PLL1

Set and reset by software.

This bit is used to enable or disable the injection of a dithering noise into the SSCG modulator. This dithering noise is generated using a rectangular probability density function.

0: Dithering noise injection enabled (default after reset)

1: Dithering noise injection disabled

Bit 13 **TPDFNDIS**: Dithering TPDF noise control for PLL1

Set and reset by software.

This bit is used to enable or disable the injection of a dithering noise into the SSCG modulator. This dithering noise is generated using a triangular probability density function.

0: Dithering noise injection enabled (default after reset)

1: Dithering noise injection disabled

Bits 12:0 **MODPER[12:0]**: Modulation Period Adjustment for PLL1

Set and reset by software to adjust the modulation period of the clock spreading generator.

7.8.16 RCC PLL2 dividers configuration register 1 (RCC_PLL2DIVR1)

Address offset: 0x038

Reset value: 0x0101 0280

Access: Word and half-word only

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

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **DIVR[6:0]**: PLL2 DIVR division factor

Set and reset by software to control the frequency of the **pll2_r_ck** clock.
0000000: **pll2_r_ck** = **vco2_ck**
0000001: **pll2_r_ck** = **vco2_ck** / 2 (default after reset)
0000010: **pll2_r_ck** = **vco2_ck** / 3
0000011: **pll2_r_ck** = **vco2_ck** / 4
...
1111111: **pll2_r_ck** = **vco2_ck** / 128

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **DIVQ[6:0]**: PLL2 DIVQ division factor

Set and reset by software to control the frequency of the **pll2_q_ck** clock.
0000000: **pll2_q_ck** = **vco2_ck**
0000001: **pll2_q_ck** = **vco2_ck** / 2 (default after reset)
0000010: **pll2_q_ck** = **vco2_ck** / 3
0000011: **pll2_q_ck** = **vco2_ck** / 4
...
1111111: **pll2_q_ck** = **vco2_ck** / 128

Bits 15:9 **DIVP[6:0]**: PLL2 DIVP division factor

Set and reset by software to control the frequency of the **pll2_p_ck** clock.
0000000: **pll2_p_ck** = **vco2_ck**
0000001: **pll2_p_ck** = **vco2_ck** / 2 (default after reset)
0000010: **pll2_p_ck** = **vco2_ck** / 3
0000011: **pll2_p_ck** = **vco2_ck** / 4
...
1111111: **pll2_p_ck** = **vco2_ck** / 128

Bits 8:0 **DIVN[8:0]**: multiplication factor for PLL2 VCO

Set and reset by software to control the multiplication factor of the VCO.

These bits can be written only when the PLL is disabled (PLL2ON = 0 in *RCC source control register (RCC_CR)*).

.....: Not used

0x006: wrong configuration

0x007: DIVN = 8

...

0x080: DIVN = 129 (default after reset)

...

0x1A3: DIVN = 420

Others: wrong configurations

Caution: The software must set correctly these bits to insure that the VCO output frequency is within its valid frequency range, that is:

– 128 to 544 MHz if PLL2VCOSEL = 0

– 150 to 420 MHz if PLL2VCOSEL = 1

VCO output frequency = F_{ref2_ck} x DIVN, when fractional value 0 has been loaded into FRACN, with

– DIVN between 8 and 420

– The input frequency F_{ref2_ck} must be between 1 and 16 MHz.

7.8.17 RCC PLL2 dividers configuration register 2 (RCC_PLL2DIVR2)

Address offset: 0x0C4

Reset value: 0x0000 0101

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

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **DIVT[2:0]**: PLL2 DIVT division factor

Set and reset by software to control the frequency of the **pll2_t_ck** clock.

This post-divider performs divisions with 50% duty-cycle.

The duty-cycle of 50% is guaranteed only in the following conditions:

– With VCOL, if (DIVT+1) is even,

– With VCOH, for all DIVT values

These bits can be written only when the PLL2TEN = 0.

000: **pll2_t_ck** = **vco2_ck**

001: **pll2_t_ck** = **vco2_ck** / 2 (default after reset)

010: **pll2_t_ck** = **vco2_ck** / 3

011: **pll2_t_ck** = **vco2_ck** / 4

100: **pll2_t_ck** = **vco2_ck** / 5

101: **pll2_t_ck** = **vco2_ck** / 6

110: **pll2_t_ck** = **vco2_ck** / 7

111: **pll2_t_ck** = **vco2_ck** / 8

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

Bits 2:0 **DIVS[2:0]**: PLL2 DIVS division factor

Set and reset by software to control the frequency of the **pll2_s_ck** clock.
This post-divider performs divisions with 50% duty-cycle.
The duty-cycle of 50% is guaranteed only in the following conditions:

- With VCOL, if (DIVS+1) is even,
- With VCOH, for all DIVS values

These bits can be written only when the PLL2SEN = 0.
000: **pll2_s_ck** = **vco2_ck**
001: **pll2_s_ck** = **vco2_ck** / 2 (default after reset)
010: **pll2_s_ck** = **vco2_ck** / 3
011: **pll2_s_ck** = **vco2_ck** / 4
100: **pll2_s_ck** = **vco2_ck** / 5
101: **pll2_s_ck** = **vco2_ck** / 6
110: **pll2_s_ck** = **vco2_ck** / 7
111: **pll2_s_ck** = **vco2_ck** / 8

7.8.18 RCC PLL2 fractional divider register (RCC_PLL2FRACR)

Address offset: 0x03C

Reset value: 0x0000 0000

Access: Word and half-word only

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:3 **FRACN[12:0]**: fractional part of the multiplication factor for PLL2 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO. These bits can be written at any time, allowing dynamic fine-tuning of the PLL2 VCO.

Caution: The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is:

- 128 to 544 MHz if PLL2VCOSEL = 0
- 150 to 420 MHz if PLL2VCOSEL = 1

VCO output frequency = F_{ref2_ck} x $\left(DIVN + (FRACN / 2^{13})\right)$, with

- DIVN between 8 and 420
- FRACN can be between 0 and 2^{13} - 1
- The input frequency F_{ref2_ck} must be between 1 and 16 MHz.

In order to change the FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:

- Set the bit PLL2FRACLEN to 0.
- Write the new fractional value into FRACN.
- Set the bit PLL2FRACLEN to 1.
- Wait at least 5 µs.

Bits 2:0 Reserved, must be kept at reset value.

7.8.19 RCC PLL2 spread spectrum clock generator register (RCC_PLL2SSCGR)

Address offset: 0x0D0

Reset value: 0x0000 0000

Access: Word and half-word only

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

Bit 31 Reserved, must be kept at reset value.

Bits 30:16 **INCSTEP[14:0]**: Modulation depth adjustment for PLL2

Set and reset by software to adjust the modulation depth of the clock spreading generator.

Bit 15 **SPREADSEL**: Spread spectrum clock generator mode for PLL2

Set and reset by software to select the clock spreading mode.

0: Center-spread modulation selected (default after reset)

1: Down-spread modulation selected

Bit 14 **RPDFNDIS**: Dithering RPDF noise control for PLL2

Set and reset by software.

This bit is used to enable or disable the injection of a dithering noise into the SSCG modulator. This dithering noise is generated using a rectangular probability density function.

0: Dithering noise injection enabled (default after reset)

1: Dithering noise injection disabled

Bit 13 **TPDFNDIS**: Dithering TPDF noise control for PLL2

Set and reset by software.

This bit is used to enable or disable the injection of a dithering noise into the SSCG modulator. This dithering noise is generated using a triangular probability density function.

0: Dithering noise injection enabled (default after reset)

1: Dithering noise injection disabled

Bits 12:0 **MODPER[12:0]**: Modulation Period Adjustment for PLL2

Set and reset by software to adjust the modulation period of the clock spreading generator.

7.8.20 RCC PLL3 dividers configuration register 1 (RCC_PLL3DIVR1)

Address offset: 0x040

Reset value: 0x0101 0280

Access: Word and half-word only.

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

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **DIVR[6:0]**: PLL3 DIVR division factor

Set and reset by software to control the frequency of the **pll3_r_ck** clock.

0000000: **pll3_r_ck** = **vco3_ck**

0000001: **pll3_r_ck** = **vco3_ck** / 2 (default after reset)

0000010: **pll3_r_ck** = **vco3_ck** / 3

0000011: **pll3_r_ck** = **vco3_ck** / 4

...

1111111: **pll3_r_ck** = **vco3_ck** / 128

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **DIVQ[6:0]**: PLL3 DIVQ division factor

Set and reset by software to control the frequency of the **pll3_q_ck** clock.

0000000: **pll3_q_ck** = **vco3_ck**

0000001: **pll3_q_ck** = **vco3_ck** / 2 (default after reset)

0000010: **pll3_q_ck** = **vco3_ck** / 3

0000011: **pll3_q_ck** = **vco3_ck** / 4

...

1111111: **pll3_q_ck** = **vco3_ck** / 128

Bits 15:9 **DIVP[6:0]**: PLL3 DIVP division factor

Set and reset by software to control the frequency of the **pll3_p_ck** clock.
0000000: **pll3_p_ck** = **vco3_ck**
0000001: **pll3_p_ck** = **vco3_ck** / 2 (default after reset)
0000010: **pll3_p_ck** = **vco3_ck** / 3
0000011: **pll3_p_ck** = **vco3_ck** / 4
...
1111111: pll3_p_ck = **vco3_ck** / 128

Bits 8:0 **DIVN[8:0]**: Multiplication factor for PLL3 VCO

Set and reset by software to control the multiplication factor of the VCO.
These bits can be written only when the PLL is disabled (PLL3ON = 0 in *RCC source control register (RCC_CR)*).
.....: Not used
0x006: Wrong configuration
0x007: DIVN = 8
...
0x080: DIVN = 129 (default after reset)
...
0x1A3: DIVN = 420
Others: Wrong configurations

Caution: The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is:

– 128 to 544 MHz if PLL3VCOSEL = 0
– 150 to 420 MHz if PLL3VCOSEL = 1

VCO output frequency = F_{ref3_ck} x DIVN, when fractional value 0 has been loaded into FRACN, with:

– DIVN between 8 and 420
– The input frequency F_{ref3_ck} must be between 1 and 16 MHz

7.8.21 RCC PLL3 dividers configuration register 2 (RCC_PLL3DIVR2)

Address offset: 0x0C8

Reset value: 0x0000 0101

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

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

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

Bits 2:0 **DIVS[2:0]**: PLL3 DIVS division factor

Set and reset by software to control the frequency of the **pll3_s_ck** clock.
This post-divider performs divisions with 50% duty-cycle.
The duty-cycle of 50% is guaranteed only in the following conditions:

- With VCOL, if (DIVS+1) is even,
- With VCOH, for all DIVS values

These bits can be written only when the PLL3SEN = 0.
000: **pll3_s_ck** = **vco3_ck**
001: **pll3_s_ck** = **vco3_ck** / 2 (default after reset)
010: **pll3_s_ck** = **vco3_ck** / 3
011: **pll3_s_ck** = **vco3_ck** / 4
100: **pll3_s_ck** = **vco3_ck** / 5
101: **pll3_s_ck** = **vco3_ck** / 6
110: **pll3_s_ck** = **vco3_ck** / 7
111: **pll3_s_ck** = **vco3_ck** / 8

7.8.22 RCC PLL3 fractional divider register (RCC_PLL3FRACR)

Address offset: 0x044

Reset value: 0x0000 0000

Access: Word and half-word only

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:3 **FRACN[12:0]**: fractional part of the multiplication factor for PLL3 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO. These bits can be written at any time, allowing dynamic fine-tuning of the PLL3 VCO.

Caution: The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is:

- 128 to 544 MHz if PLL3VCOSEL = 0
- 150 to 420 MHz if PLL3VCOSEL = 1

VCO output frequency = F_{ref3_ck} x $\left(DIVN + (FRACN / 2^{13})\right)$, with

- DIVN between 8 and 420
- FRACN can be between 0 and 2^{13} - 1
- The input frequency F_{ref3_ck} must be between 1 and 16 MHz.

In order to change the FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:

- Set the bit PLL1FRACEN to 0.
- Write the new fractional value into FRACN.
- Set the bit PLL1FRACEN to 1
- Wait at least 5 µs.

Bits 2:0 Reserved, must be kept at reset value.

7.8.23 RCC PLL3 spread spectrum clock generator register (RCC_PLL3SSCGR)

Address offset: 0x0D4

Reset value: 0x0000 0000

Access: Word and half-word only

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

Bit 31 Reserved, must be kept at reset value.

Bits 30:16 **INCSTEP[14:0]**: Modulation depth adjustment for PLL3

Set and reset by software to adjust the modulation depth of the clock spreading generator.

Bit 15 **SPREADSEL**: Spread spectrum clock generator mode for PLL3

Set and reset by software to select the clock spreading mode.

0: Center-spread modulation selected (default after reset)

1: Down-spread modulation selected

Bit 14 **RPDFNDIS**: Dithering RPDF noise control for PLL3

Set and reset by software.

This bit is used to enable or disable the injection of a dithering noise into the SSCG modulator. This dithering noise is generated using a rectangular probability density function.

0: Dithering noise injection enabled (default after reset)

1: Dithering noise injection disabled

Bit 13 **TPDFNDIS**: Dithering TPDF noise control for PLL3

Set and reset by software.

This bit is used to enable or disable the injection of a dithering noise into the SSCG modulator. This dithering noise is generated using a triangular probability density function.

0: Dithering noise injection enabled (default after reset)

1: Dithering noise injection disabled

Bits 12:0 **MODPER[12:0]**: Modulation Period Adjustment for PLL3

Set and reset by software to adjust the modulation period of the clock spreading generator.

7.8.24 RCC peripheral kernel clock selection register 1 (RCC_CCIPR1)

Address offset: 0x04C

Reset value: 0x0000 0000

Changing the clock source on-the-fly is allowed (except for CKPERSEL) and does not generate any timing violation. However the user must make sure that both the previous and the new clock sources are present during the switching, and during the whole transition time. Refer to *Clock switches and gating*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CKPERSEL[1:0]		Res.	ADCSEL[2:0]			Res.	ADF1SEL[2:0]			ADCDIV[3:0]			
		rw	rw		rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OTGFSSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XSPI1SEL[1:0]		Res.	SDMMC1SEL	FMCSEL[1:0]	
rw	rw									rw	rw		rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **CKPERSEL[1:0]**: **per_ck** clock source selection

00: **hsi_ker_ck** selected as **per_ck** clock (default after reset)

01: **csi_ker_ck** selected as **per_ck** clock

10: **hse_ker_ck** selected as **per_ck** clock

11: Reserved, the **ck_per** clock is disabled

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **ADCSEL[2:0]**: SAR ADC kernel clock source selection
Set and reset by software.
000: **hclk3** selected as kernel peripheral clock (default after reset)
001: **rcc_timx_ker_ck** selected as kernel peripheral clock
010: **hsi_ker_ck** selected as kernel peripheral clock
011: **hse_ker_ck** selected as kernel peripheral clock
100: **pll2_p_ck** selected as kernel peripheral clock
101: **pll3_r_ck** selected as kernel peripheral clock
110: **per_ck** selected as kernel peripheral clock
111: Reserved, the kernel peripheral clock is disabled

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **ADF1SEL[2:0]**: ADF kernel clock source selection
Set and reset by software.
000: **hclk1** selected as ADF kernel clock (default after reset)
001: **pll2_p_ck** selected as ADF kernel clock
010: **pll3_p_ck** selected as ADF kernel clock
011: **AUDIOCLK** selected as ADF kernel clock
100: **csi_ker_ck** selected as ADF kernel clock
101: **hsi_ker_ck** selected as ADF kernel clock
Others: Reserved, the kernel clock is disabled
Note: AUDIOCLK is an external clock taken from a pin.

Bits 19:16 **ADCDIV[3:0]**: ADC kernel clock division factor
0000: ADC kernel clock not divided (default after reset)
0001: ADC kernel clock divided by 2
0010: ADC kernel clock divided by 4
0011: ADC kernel clock divided by 6
0100: ADC kernel clock divided by 8
0101: ADC kernel clock divided by 10
0110: ADC kernel clock divided by 12
0111: ADC kernel clock divided by 16
1000: ADC kernel clock divided by 32
1001: ADC kernel clock divided by 64
1010: ADC kernel clock divided by 128
1011: ADC kernel clock divided by 256
Others: Reserved

Bits 15:14 **OTGFSSEL[1:0]**: OTGFS kernel clock source selection
Set and reset by software.
00: hsi48_ker_ck (default after reset)
01: pll3_q_ck
10: hse_ker_ck
11: Reserved, the kernel clock is disabled

Bits 13:6 Reserved, must be kept at reset value.

Bits 5:4 **XSPI1SEL[1:0]**: XSPI1 kernel clock source selection
Set and reset by software.
00: **hclk5** selected as kernel peripheral clock (default after reset)
01: **pll2_s_ck** selected as kernel peripheral clock
1x: **pll2_t_ck** selected as kernel peripheral clock

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SDMMC1SEL**: SDMMC1 kernel clock source selection
Set and reset by software.
0: **pll2_s_ck** selected as kernel peripheral clock (default after reset)
1: **pll2_t_ck** selected as kernel peripheral clock

Bits 1:0 **FMCSEL[1:0]**: FMC kernel clock source selection
Set and reset by software.
00: **hclk5** selected as kernel peripheral clock (default after reset)
01: **pll1_q_ck** selected as kernel peripheral clock
10: **pll2_r_ck** selected as kernel peripheral clock
11: **hsi_ker_ck** selected as kernel peripheral clock

7.8.25 RCC peripheral kernel clock selection register 2 (RCC_CCIPR2)

Address offset: 0x050

Reset value: 0x0000 0000

Changing the clock source on-the-fly is allowed and does not generate any timing violation. However the user must make sure that both the previous and the new clock sources are present during the switching, and during the whole transition time. Refer to *Clock switches and gating*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANSEL[1:0]		Res.	Res.	Res.	LPTIM1SEL[2:0]		
								rw	rw				rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	I3C1SEL[1:0]		Res.	Res.	I2C123SEL[1:0]		Res.	SPI23SEL[2:0]			Res.	UART234578SEL[2:0]		
		rw	rw			rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **FDCANSEL[1:0]**: FDCAN kernel clock source selection
00: **hse_ker_ck** selected as FDCAN kernel clock (default after reset)
01: **pll1_q_ck** selected as FDCAN kernel clock
10: **pll2_p_ck** selected as FDCAN kernel clock
11: Reserved, the kernel clock is disabled

Bits 21:19 Reserved, must be kept at reset value.

Bits 18:16 **LPTIM1SEL[2:0]**: LPTIM1 kernel clock source selection
Set and reset by software.
000: **pclk1** selected as kernel peripheral clock (default after reset)
001: **pll2_p_ck** selected as kernel peripheral clock
010: **pll3_r_ck** selected as kernel peripheral clock
011: **lse_ck** selected as kernel peripheral clock
100: **lsi_ck** selected as kernel peripheral clock
101: **per_ck** selected as kernel peripheral clock
Others: Reserved, the kernel clock is disabled

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

Bits 13:12 **I3C1SEL[1:0]**: I3C1 kernel clock source selection
Set and reset by software.
00: **pclk1** selected as kernel peripheral clock (default after reset)
01: **pll3_r_ck** selected as kernel peripheral clock
10: **hsi_ker_ck** selected as kernel peripheral clock
11: **csi_ker_ck** selected as kernel peripheral clock

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **I2C123SEL[1:0]**: I2C1, I2C2, I2C3 kernel clock source selection
Set and reset by software.
00: **pclk1** selected as kernel clock (default after reset)
01: **pll3_r_ck** selected as kernel clock
10: **hsi_ker_ck** selected as kernel clock
11: **csi_ker_ck** selected as kernel clock

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **SPI23SEL[2:0]**: SPI/I2S2 and SPI/I2S3 kernel clock source selection
Set and reset by software.

Caution: If the selected clock is the external clock and this clock is stopped, it is not be possible to switch to another clock. Refer to *Clock switches and gating on page 378* for additional information.

000: **pll1_q_ck** selected as kernel clock (default after reset)
001: **pll2_p_ck** selected as kernel clock
010: **pll3_p_ck** selected as kernel clock
011: **AUDIOCLK** selected as kernel clock
100: **per_ck** selected as kernel clock
Others: Reserved, the kernel clock is disabled

Note: AUDIOCLK is an external clock taken from a pin.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **UART234578SEL[2:0]**: USART2,3, UART4,5,7,8 (APB1) kernel clock source selection
Set and reset by software.
000: **pclk1** selected as kernel clock (default after reset)
001: **pll2_q_ck** selected as kernel clock
010: **pll3_q_ck** selected as kernel clock
011: **hsi_ker_ck** selected as kernel clock
100: **csi_ker_ck** selected as kernel clock
101: **lse_ck** selected as kernel clock
Others: Reserved, the kernel clock is disabled

7.8.26 RCC peripheral kernel clock selection register 3 (RCC_CCIPR3)

Address offset: 0x054

Reset value: 0x0000 0000

Changing the clock source on-the-fly is allowed and does not generate any timing violation. However the user must make sure that both the previous and the new clock sources are present during the switching, and during the whole transition time. Refer to *Clock switches and gating*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SPI1SEL[2:0]			Res.	SPI45SEL[2:0]			Res.	USART16910SEL[2:0]		
					rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **SPI1SEL[2:0]**: SPI/I2S1 kernel clock source selection

Set and reset by software.

Caution: If the selected clock is the external clock (AUDIOCLK) and this clock is stopped, it is not possible to switch to another clock. Refer to *Clock switches and gating on page 378* for additional information.

000: **pll1_q_ck** selected as SPI/I2S1 kernel clock (default after reset)
001: **pll2_p_ck** selected as SPI/I2S1 kernel clock
010: **pll3_p_ck** selected as SPI/I2S1 kernel clock
011: **AUDIOCLK** selected as SPI/I2S1 kernel clock
100: **per_ck** selected as SPI/I2S1 kernel clock
Others: Reserved, the kernel clock is disabled

Note: I2S_CKIN is an external clock taken from a pin.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **SPI45SEL[2:0]**: SPI4 and 5 kernel clock source selection

Set and reset by software.
000: **pclk2** selected as kernel clock (default after reset)
001: **pll2_q_ck** is selected as kernel clock
010: **pll3_q_ck** is selected as kernel clock
011: **hsi_ker_ck** is selected as kernel clock
100: **csi_ker_ck** is selected as kernel clock
101: **hse_ker_ck** is selected as kernel clock
Others: Reserved, the kernel clock is disabled

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **USART16910SEL[2:0]**: USART1, UARTS6, USART9 and USART10 kernel clock source selection

Set and reset by software.
000: **pclk2** selected as kernel clock (default after reset)
001: **pll2_q_ck** selected as kernel clock
010: **pll3_q_ck** selected as kernel clock
011: **hsi_ker_ck** selected as kernel clock
100: **csi_ker_ck** selected as kernel clock
101: **lse_ck** selected as kernel clock
Others: Reserved, the kernel clock is disabled

7.8.27 RCC peripheral kernel clock selection register 4 (RCC_CCIPR4)

Address offset: 0x058

Reset value: 0x0000 0000

Changing the clock source on-the-fly is allowed and does not generate any timing violation. However the user must make sure that both the previous and the new clock sources are

present during the switching, and during the whole transition time. Refer to *Clock switches and gating*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DAC4SEL[2:0]			Res.	DAC3SEL[2:0]			Res.	DAC2SEL[2:0]			DAC1SHSEL	DAC1SEL[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LPTIM45SEL[2:0]			Res.	LPTIM23SEL[2:0]			Res.	SPI6SEL[2:0]			Res.	LPUART1SEL[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **DAC4SEL[2:0]**: DAC4 kernel clock source selection
Set and reset by software.
000: **hclk3** selected as kernel peripheral clock - synchronous mode (default after reset)
001: **rcc_tim_ker_ck** selected as kernel peripheral clock
010: **rcc_adc_ker_ck** selected as kernel peripheral clock
011: **hsi_ker_ck** selected as kernel peripheral clock
100: **hse_ker_ck** selected as kernel peripheral clock
101: **pll2_p_ck** selected as kernel peripheral clock
110: **pll3_r_ck** selected as kernel peripheral clock
111: **per_ck** selected as kernel peripheral clock

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **DAC3SEL[2:0]**: DAC3 kernel clock source selection
Set and reset by software.
000: **hclk3** selected as kernel peripheral clock - synchronous mode (default after reset)
001: **rcc_tim_ker_ck** selected as kernel peripheral clock
010: **rcc_adc_ker_ck** selected as kernel peripheral clock
011: **hsi_ker_ck** selected as kernel peripheral clock
100: **hse_ker_ck** selected as kernel peripheral clock
101: **pll2_p_ck** selected as kernel peripheral clock
110: **pll3_r_ck** selected as kernel peripheral clock
111: **per_ck** selected as kernel peripheral clock

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **DAC2SEL[2:0]**: DAC2 kernel clock source selection
Set and reset by software.
000: **hclk3** selected as kernel peripheral clock - synchronous mode (default after reset)
001: **rcc_tim_ker_ck** selected as kernel peripheral clock
010: **rcc_adc_ker_ck** selected as kernel peripheral clock
011: **hsi_ker_ck** selected as kernel peripheral clock
100: **hse_ker_ck** selected as kernel peripheral clock
101: **pll2_p_ck** selected as kernel peripheral clock
110: **pll3_r_ck** selected as kernel peripheral clock
111: **per_ck** selected as kernel peripheral clock

Bit 19 **DAC1SHSEL**: DAC1 hold clock selection
Set and reset by software.
0: **lse_ck** selected as rcc_dac_hold_ck
1: **lsi_ck** selected as rcc_dac_hold_ck

Bits 18:16 **DAC1SEL[2:0]**: DAC1 kernel clock source selection
Set and reset by software.
000: **hclk3** selected as kernel peripheral clock - synchronous mode (default after reset)
001: **rcc_tim_ker_ck** selected as kernel peripheral clock
010: **rcc_adc_ker_ck** selected as kernel peripheral clock
011: **hsi_ker_ck** selected as kernel peripheral clock
100: **hse_ker_ck** selected as kernel peripheral clock
101: **pll2_p_ck** selected as kernel peripheral clock
110: **pll3_r_ck** selected as kernel peripheral clock
111: **per_ck** selected as kernel peripheral clock

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **LPTIM45SEL[2:0]**: LPTIM4, and LPTIM5 kernel clock source selection
Set and reset by software.
000: **pclk4** selected as kernel peripheral clock (default after reset)
001: **pll2_p_ck** selected as kernel peripheral clock
010: **pll3_r_ck** selected as kernel peripheral clock
011: **lse_ck** selected as kernel peripheral clock
100: **lsi_ck** selected as kernel peripheral clock
101: **per_ck** selected as kernel peripheral clock
Others: Reserved, the kernel clock is disabled

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **LPTIM23SEL[2:0]**: LPTIM2 and LPTIM3 kernel clock source selection
Set and reset by software.
000: **pclk4** selected as kernel peripheral clock (default after reset)
001: **pll2_p_ck** selected as kernel peripheral clock
010: **pll3_r_ck** selected as kernel peripheral clock
011: **lse_ck** selected as kernel peripheral clock
100: **lsi_ck** selected as kernel peripheral clock
101: **per_ck** selected as kernel peripheral clock
Others: Reserved, the kernel clock is disabled

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **SPI6SEL[2:0]**: SPI/I2S6 kernel clock source selection
Set and reset by software.
000: **pclk4** selected as kernel peripheral clock (default after reset)
001: **pll2_q_ck** selected as kernel peripheral clock
010: **pll3_q_ck** selected as kernel peripheral clock
011: **hsi_ker_ck** selected as kernel peripheral clock
100: **csi_ker_ck** selected as kernel peripheral clock
101: **hse_ker_ck** selected as kernel peripheral clock
Others: Reserved, the kernel clock is disabled

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **LPUART1SEL[2:0]**: LPUART1 kernel clock source selection
Set and reset by software.
000: **pclk4** selected as kernel peripheral clock (default after reset)
001: **pll2_q_ck** selected as kernel peripheral clock
010: **pll3_q_ck** selected as kernel peripheral clock
011: **hsi_ker_ck** selected as kernel peripheral clock
100: **csi_ker_ck** selected as kernel peripheral clock
101: **lse_ck** selected as kernel peripheral clock
Others: Reserved, the kernel clock is disabled

7.8.28 RCC clock source interrupt enable register (RCC_CIER)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	LSECSSIE	PLL3RDYIE	PLL2RDYIE	PLL1RDYIE	HSI48RDYIE	CSIRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 9 **LSECSSIE**: LSE clock security system interrupt enable
Set and reset by software to enable/disable interrupt caused by the clock security system (CSS) on external 32 kHz oscillator.
0: LSE CSS interrupt disabled (default after reset)
1: LSE CSS interrupt enabled

Bit 8 **PLL3RDYIE**: PLL3 ready interrupt enable
Set and reset by software to enable/disable interrupt caused by PLL3 lock.
0: PLL3 lock interrupt disabled (default after reset)
1: PLL3 lock interrupt enabled

Bit 7 **PLL2RDYIE**: PLL2 ready interrupt enable
Set and reset by software to enable/disable interrupt caused by PLL2 lock.
0: PLL2 lock interrupt disabled (default after reset)
1: PLL2 lock interrupt enabled

Bit 6 **PLL1RDYIE**: PLL1 ready interrupt enable
Set and reset by software to enable/disable interrupt caused by PLL1 lock.
0: PLL1 lock interrupt disabled (default after reset)
1: PLL1 lock interrupt enabled

Bit 5 **HSI48RDYIE**: HSI48 ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the HSI48 oscillator stabilization.
0: HSI48 ready interrupt disabled (default after reset)
1: HSI48 ready interrupt enabled

Bit 4 **CSIRDYIE**: CSI ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the CSI oscillator stabilization.
0: CSI ready interrupt disabled (default after reset)
1: CSI ready interrupt enabled

Bit 3 **HSERDYIE**: HSE ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the HSE oscillator stabilization.
0: HSE ready interrupt disabled (default after reset)
1: HSE ready interrupt enabled

Bit 2 **HSIRDYIE**: HSI ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the HSI oscillator stabilization.
0: HSI ready interrupt disabled (default after reset)
1: HSI ready interrupt enabled

Bit 1 **LSERDYIE**: LSE ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the LSE oscillator stabilization.
0: LSE ready interrupt disabled (default after reset)
1: LSE ready interrupt enabled

Bit 0 **LSIRDYIE**: LSI ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the LSI oscillator stabilization.
0: LSI ready interrupt disabled (default after reset)
1: LSI ready interrupt enabled

7.8.29 RCC clock source interrupt flag register (RCC_CIFR)

Address offset: 0x64

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSECSSF	LSECSSF	PLL3RDYF	PLL2RDYF	PLL1RDYF	HSI48RDYF	CSIRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **HSECSSF**: HSE clock security system interrupt flag
Reset by software by writing HSECSSC bit.
Set by hardware in case of HSE clock failure.
0: No clock security interrupt caused by HSE clock failure (default after reset)
1: Clock security interrupt caused by HSE clock failure

Bit 9 **LSECSSF**: LSE clock security system interrupt flag
Reset by software by writing LSECSSC bit.
Set by hardware when a failure is detected on the external 32 kHz oscillator and LSECSSIE is set.
0: No failure detected on the external 32 kHz oscillator (default after reset)
1: Failure detected on the external 32 kHz oscillator

Bit 8 **PLL3RDYF**: PLL3 ready interrupt flag
Reset by software by writing PLL3RDYC bit.
Set by hardware when the PLL3 locks and PLL3RDYIE is set.
0: No clock ready interrupt caused by PLL3 lock (default after reset)
1: Clock ready interrupt caused by PLL3 lock

Bit 7 **PLL2RDYF**: PLL2 ready interrupt flag
Reset by software by writing PLL2RDYC bit.
Set by hardware when the PLL2 locks and PLL2RDYIE is set.
0: No clock ready interrupt caused by PLL2 lock (default after reset)
1: Clock ready interrupt caused by PLL2 lock

Bit 6 **PLL1RDYF**: PLL1 ready interrupt flag
Reset by software by writing PLL1RDYC bit.
Set by hardware when the PLL1 locks and PLL1RDYIE is set.
0: No clock ready interrupt caused by PLL1 lock (default after reset)
1: Clock ready interrupt caused by PLL1 lock

Bit 5 **HSI48RDYF**: HSI48 ready interrupt flag
Reset by software by writing HSI48RDYC bit.
Set by hardware when the HSI48 clock becomes stable and HSI48RDYIE is set.
0: No clock ready interrupt caused by the HSI48 oscillator (default after reset)
1: Clock ready interrupt caused by the HSI48 oscillator

Bit 4 **CSIRDYF**: CSI ready interrupt flag
Reset by software by writing CSIRDYC bit.
Set by hardware when the CSI clock becomes stable and CSIRDYIE is set.
0: No clock ready interrupt caused by the CSI (default after reset)
1: Clock ready interrupt caused by the CSI

Bit 3 **HSERDYF**: HSE ready interrupt flag
Reset by software by writing HSERDYC bit.
Set by hardware when the HSE clock becomes stable and HSERDYIE is set.
0: No clock ready interrupt caused by the HSE (default after reset)
1: Clock ready interrupt caused by the HSE

Bit 2 **HSIRDYF**: HSI ready interrupt flag
Reset by software by writing HSIRDYC bit.
Set by hardware when the HSI clock becomes stable and HSIRDYIE is set.
0: No clock ready interrupt caused by the HSI (default after reset)
1: Clock ready interrupt caused by the HSI

Bit 1 **LSERDYF**: LSE ready interrupt flag
Reset by software by writing LSERDYC bit.
Set by hardware when the LSE clock becomes stable and LSERDYIE is set.
0: No clock ready interrupt caused by the LSE (default after reset)
1: Clock ready interrupt caused by the LSE

Bit 0 **LSIRDYF**: LSI ready interrupt flag
Reset by software by writing LSIRDYC bit.
Set by hardware when the LSI clock becomes stable and LSIRDYIE is set.
0: No clock ready interrupt caused by the LSI (default after reset)
1: Clock ready interrupt caused by the LSI

7.8.30 RCC clock source interrupt clear register (RCC_CICR)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSECSSC	LSECSSC	PLL3RDYC	PLL2RDYC	PLL1RDYC	HSI48RDYC	CSIRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC
					w	w	w	w	w	w	w	w	w	w	w

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **HSECSSC**: HSE clock security system interrupt clear
Set by software to clear HSECSSF.
Reset by hardware when clear done.
0: No effect
1: Clear HSECSSF

Bit 9 **LSECSSC**: LSE clock security system interrupt clear
Set by software to clear LSECSSF.
Reset by hardware when clear done.
0: No effect
1: Clear LSECSSF

Bit 8 **PLL3RDYC**: PLL3 ready interrupt clear
Set by software to clear PLL3RDYF.
Reset by hardware when clear done.
0: No effect
1: Clear PLL3RDYF

Bit 7 **PLL2RDYC**: PLL2 ready interrupt clear
Set by software to clear PLL2RDYF.
Reset by hardware when clear done.
0: No effect
1: Clear PLL2RDYF

Bit 6 **PLL1RDYC**: PLL1 ready interrupt clear
Set by software to clear PLL1RDYF.
Reset by hardware when clear done.
0: No effect
1: Clear PLL1RDYF

Bit 5 **HSI48RDYC**: HSI48 ready interrupt clear
Set by software to clear HSI48RDYF.
Reset by hardware when clear done.
0: No effect
1: Clear HSI48RDYF

Bit 4 **CSIRDYC**: CSI ready interrupt clear
Set by software to clear CSIRDYF.
Reset by hardware when clear done.
0: No effect
1: Clear CSIRDYF

Bit 3 **HSERDYC**: HSE ready interrupt clear
Set by software to clear HSERDYF.
Reset by hardware when clear done.
0: No effect
1: Clear HSERDYF

Bit 2 **HSIRDYC**: HSI ready interrupt clear
Set by software to clear HSIRDYF.
Reset by hardware when clear done.
0: No effect
1: Clear HSIRDYF

Bit 1 **LSERDYC**: LSE ready interrupt clear
Set by software to clear LSERDYF.
Reset by hardware when clear done.
0: No effect
1: Clear LSERDYF

Bit 0 **LSIRDYC**: LSI ready interrupt clear
Set by software to clear LSIRDYF.
Reset by hardware when clear done.
0: No effect
1: Clear LSIRDYF

7.8.31 RCC Backup domain control register (RCC_BDCR)

Address offset: 0x070

Reset value: 0x0000 0010

Reset by Backup domain reset.

Access: 0 ≤ wait state ≤ 7, word, half-word and byte access. Wait states are inserted in case of successive accesses to this register.

After a system reset, the RCC_BDCR register is write-protected. To modify this register, the DBP bit in the *PWR control register 1 (PWR_CR1)* must be set to 1. RCC_BDCR bits are only reset after a Backup domain reset (see *Section 7.4.5: Backup domain reset*). Any other internal or external reset does not have any effect on these bits.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Res.	Res.	LSECSSRA	Res.	Res.	RTCSEL[1:0]		LSEEXT	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
rw			rw			rwo	rwo	rw	r	rw	rw	rw	rw	r	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **VSWRST:** VSwitch domain software reset

Set and reset by software.
To generate a V_{SW} reset, it is recommended to write this bit to 1, then back to 0.
0: Reset not activated (default after Backup domain reset)
1: Generates a reset pulse, resetting the entire V_{SW} domain.

Bit 15 **RTCEN**: RTC clock enable

Set and reset by software.
0: **rtc_ck** disabled (default after Backup domain reset)
1: **rtc_ck** enabled

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **LSECSSRA**: Re-Arm the LSECSS function

Set by software.
After a LSE failure detection, the software application can re-enable the LSECSS by writing this bit to 1. Reading this bit returns the written value. Prior to set this bit to 1, LSECSSON must be set to 0. Refer to *Section : CSS on LSE* for details.
0: Writing 0 has no effect (default after Backup domain reset)
1: Writing 1 generates a re-arm pulse for the LSECSS function

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **RTCSEL[1:0]**:RTC clock source selection

Set by software to select the clock source for the RTC. These bits can be written only one time (except in case of failure detection on LSE). These bits must be written before LSECSSON is enabled. The VSWRST bit can be used to reset them, then it can be written one time again.

If HSE is selected as RTC clock, this clock is lost when the system is in Stop mode or in case of a pin reset (NRST).

00: No clock (default after Backup domain reset)

01: LSE selected as RTC clock

10: LSI selected as RTC clock

11: HSE divided by RTCPRE value selected as RTC clock

Bit 7 **LSEEXT**: low-speed external clock enable in Bypass mode

Set by software to enable the external clock when in BYPASS mode (LSEBYP = 1).

The LSEEXT bit can be written only if the LSE oscillator is disabled (LSEON = 0).

0: External LSE disabled (default after Backup domain reset)

1: External LSE enabled.

Bit 6 **LSECSSD**: LSE clock security system failure detection

Set by hardware to indicate when a failure has been detected by the clock security system on the external 32 kHz oscillator.

0: No failure detected on 32 kHz oscillator (default after Backup domain reset)

1: Failure detected on 32 kHz oscillator

Bit 5 **LSECSSON**: LSE clock security system enable

Set by software to enable the clock security system on 32 kHz oscillator.

LSECSSON must be enabled after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware) and after RTCSEL is selected.

Once enabled, this bit can only be disabled following

- an LSE failure detection (LSECSSD = 1). In that case the software must disable LSECSSON.
- a back-up domain reset

0: CSS on 32 kHz oscillator OFF (default after Backup domain reset)

1: CSS on 32 kHz oscillator ON

Bits 4:3 **LSEDRV[1:0]**: LSE oscillator driving capability

Set by software to select the driving capability of the LSE oscillator.

00: Lowest drive

01: Medium-low drive

10: Medium-high drive (default after backup domain reset)

11: Highest drive

Bit 2 **LSEBYP**: LSE oscillator bypass
Set and reset by software to bypass oscillator in debug mode. This bit can only be written when the LSE is disabled (LSEON = 0)
0: LSE oscillator not bypassed (default after Backup domain reset)
1: LSE oscillator bypassed

Bit 1 **LSERDY**: LSE oscillator ready
Set and reset by hardware to indicate when the LSE is stable. This bit goes low six cycles of **lse_ck** clock after LSEON has been set to 0.
0: LSE oscillator not ready (default after Backup domain reset)
1: LSE oscillator ready

Bit 0 **LSEON**: LSE oscillator enabled
Set and reset by software.
0: LSE oscillator OFF (default after Backup domain reset)
1: LSE oscillator ON

7.8.32 RCC clock control and status register (RCC_CSR)

Address offset: 0x074

Reset value: 0x0000 0000

Access: 0 ≤ wait state ≤ 7, word, half-word and byte access

Wait states are inserted in case of successive accesses to this register.

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

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

Bit 1 **LSIRDY**: LSI oscillator ready
Set and reset by hardware to indicate when the low-speed internal RC oscillator is stable.
This bit needs 3 cycles of **lsi_ck** clock to fall down after LSION has been set to 0.
This bit can be set even when LSION is not enabled if there is a request for LSI clock by the clock security system on LSE or by the low-speed watchdog or by the RTC.
0: LSI clock is not ready (default after reset)
1: LSI clock is ready

Bit 0 **LSION**: LSI oscillator enable
Set and reset by software.
0: LSI is OFF (default after reset)
1: LSI is ON

7.8.33 RCC AHB1 peripheral reset register (RCC_AHB1RSTR)

Address offset: 0x080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADF1RST	Res.	Res.	Res.	OTGFSRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA1RST	HSP1RST	Res.	Res.	Res.
											rw	rw			

Bit 31 **ADF1RST**: ADF block reset
Set and reset by software.
0: Does not reset ADF block (default after reset)
1: Resets ADF block

Bits 30:28 Reserved, must be kept at reset value.

Bit 27 **OTGFSRST**: OTGFS block reset
Set and reset by software.
0: Does not reset OTGFS block (default after reset)
1: Resets OTGFS block

Bits 26:5 Reserved, must be kept at reset value.

Bit 4 **GPDMA1RST**: GPDMA1 blocks reset
Set and reset by software.
0: Does not reset GPDMA1 block (default after reset)
1: Resets GPDMA1 block

Bit 3 **HSP1RST**: HSP block reset
Set and reset by software.
0: Does not reset HSP block (default after reset)
1: Resets HSP block

Bits 2:0 Reserved, must be kept at reset value.

7.8.34 RCC AHB2 peripheral reset register (RCC_AHB2RSTR)

Address offset: 0x084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMAC1RST	CORDICRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FMAC1RST**: FMAC1 reset
Set and reset by software.
0: Does not reset FMAC1 block (default after reset)
1: Resets FMAC1 block

Bit 14 **CORDICRST**: CORDIC reset
Set and reset by software.
0: Does not reset CORDIC block (default after reset)
1: Resets CORDIC block

Bits 13:0 Reserved, must be kept at reset value.

7.8.35 RCC AHB3 peripheral reset register (RCC_AHB3RSTR)

Address offset: 0x0A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DAC4RST	DAC3RST	DAC2RST	DAC1RST	Res.	Res.	ADC34RST	ADC12RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw			rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HASHRST	RNGRST
														rw	rw

Bit 31 **DAC4RST**: DAC4block reset
Set and reset by software.
0: Does not reset DAC4 block (default after reset)
1: Resets DAC4 block

Bit 30 **DAC3RST**: DAC3 block reset
Set and reset by software.
0: Does not reset DAC3 block (default after reset)
1: Resets DAC3 block

Bit 29 **DAC2RST**: DAC2 block reset
Set and reset by software.
0: Does not reset DAC2 block (default after reset)
1: Resets DAC2 block

Bit 28 **DAC1RST**: DAC1 block reset
Set and reset by software.
0: Does not reset DAC1 block (default after reset)
1: Resets DAC1 block

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **ADC34RST**: ADC34 block reset
Set and reset by software.
0: Does not reset ADC34 block (default after reset)
1: Resets ADC34 block

Bit 24 **ADC12RST**: ADC12 block reset
Set and reset by software.
0: Does not reset ADC12 block (default after reset)
1: Resets ADC12 block

Bits 23:2 Reserved, must be kept at reset value.

Bit 1 **HASHRST**: HASH block reset
Set and reset by software.
0: Does not reset HASH block (default after reset)
1: Resets HASH block

Bit 0 **RNGRST**: random number generator block reset
Set and reset by software.
0: Does not reset RNG block (default after reset)
1: Resets RNG block

7.8.36 RCC AHB4 peripheral reset register (RCC_AHB4RSTR)

Address offset: 0x088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCRST	Res.	Res.	Res.
												rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOIRST	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **CRCRST**: CRC block reset
Set and reset by software.
0: Does not reset the CRC block (default after reset)
1: Resets the CRC block

Bits 18:9 Reserved, must be kept at reset value.

Bit 8 **GPIOIRST**: GPIOI block reset
Set and reset by software.
0: Does not reset the GPIOI block (default after reset)
1: Resets the GPIOI block

Bit 7 **GPIOHRST**: GPIOH block reset
Set and reset by software.
0: Does not reset the GPIOH block (default after reset)
1: Resets the GPIOH block

Bit 6 **GPIOGRST**: GPIOG block reset
Set and reset by software.
0: Does not reset the GPIOG block (default after reset)
1: Resets the GPIOG block

Bit 5 **GPIOFRST**: GPIOF block reset
Set and reset by software.
0: Does not reset the GPIOF block (default after reset)
1: Resets the GPIOF block

Bit 4 **GPIOERST**: GPIOE block reset
Set and reset by software.
0: Does not reset the GPIOE block (default after reset)
1: Resets the GPIOE block

Bit 3 **GPIODRST**: GPIOD block reset
Set and reset by software.
0: Does not reset the GPIOD block (default after reset)
1: Resets the GPIOD block

Bit 2 **GPIOCRST**: GPIOC block reset
Set and reset by software.
0: Does not reset the GPIOC block (default after reset)
1: Resets the GPIOC block

Bit 1 **GPIOBRST**: GPIOB block reset
Set and reset by software.
0: Does not reset the GPIOB block (default after reset)
1: Resets the GPIOB block

Bit 0 **GPIOARST**: GPIOA block reset
Set and reset by software.
0: Does not reset the GPIOA block (default after reset)
1: Resets the GPIOA block

7.8.37 RCC AHB5 peripheral reset register (RCC_AHB5RSTR)

Address offset: 0x07C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1RST	Res.	Res.	XSPI1RST	FMCRST	Res.	Res.	Res.	HPDMA1RST
							rw			rw	rw				rw

Bit 31 **NPURST**: NPU block reset
Set and reset by software.
0: Reset is released (default after reset)
1: Reset is asserted

Bits 30:9 Reserved, must be kept at reset value.

Bit 8 **SDMMC1RST**: SDMMC1 and DB_SDMMC1 blocks reset
Set and reset by software.
0: Reset is released (default after reset)
1: Reset is asserted

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **XSPI1RST**: XSPI1 and MCE1 blocks reset
Set and reset by software.
The hardware prevents writing this bit if XSPICKP = 1.
0: Reset is released (default after reset)
1: Reset is asserted

Bit 4 **FMCRST**: FMC and MCE3 blocks reset
Set and reset by software.
The hardware prevents writing this bit if FMCCKP = 1.
0: Reset is released (default after reset)
1: Reset is asserted

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **HPDMA1RST**: HPDMA1 block reset
Set and reset by software.
0: Reset is released (default after reset)
1: Reset is asserted

7.8.38 RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1)

Address offset: 0x090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8RST	UART7RST	Res.	Res.	Res.	Res.	Res.	I3C1RST	I2C3RST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Res.
rw	rw						rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3RST	SPI2RST	TIM11RST	TIM10RST	Res.	Res.	LPTIM1RST	TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UART8RST**: UART8 block reset
Set and reset by software.
0: Does not reset the UART8 block (default after reset)
1: Resets the UART8 block

Bit 30 **UART7RST**: UART7 block reset
Set and reset by software.
0: Does not reset the UART7 block (default after reset)
1: Resets the UART7 block

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **I3C1RST**: I3C1 block reset
Set and reset by software.
0: Does not reset the I3C1 block (default after reset)
1: Resets the I3C1 block

Bit 23 **I2C3RST**: I2C3 block reset
Set and reset by software.
0: Does not reset the I2C3 block (default after reset)
1: Resets the I2C3 block

Bit 22 **I2C2RST**: I2C2 block reset
Set and reset by software.
0: Does not reset the I2C2 block (default after reset)
1: Resets the I2C2 block

Bit 21 **I2C1RST**: I2C1/I3C1 block reset
Set and reset by software.
0: Does not reset the I2C1/I3C1 block (default after reset)
1: Resets the I2C1/I3C1 block

Bit 20 **UART5RST**: UART5 block reset
Set and reset by software.
0: Does not reset the UART5 block (default after reset)
1: Resets the UART5 block

Bit 19 **UART4RST**: UART4 block reset
Set and reset by software.
0: Does not reset the UART4 block (default after reset)
1: Resets the UART4 block

Bit 18 **USART3RST**: USART3 block reset
Set and reset by software.
0: Does not reset the USART3 block (default after reset)
1: Resets the USART3 block

Bit 17 **USART2RST**: USART2 block reset
Set and reset by software.
0: Does not reset the USART2 block (default after reset)
1: Resets the USART2 block

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3RST**: SPI2S3 block reset
Set and reset by software.
0: Does not reset the SPI2S3 block (default after reset)
1: Resets the SPI2S3 block

Bit 14 **SPI2RST**: SPI2S2 block reset
Set and reset by software.
0: Does not reset the SPI2S2 block (default after reset)
1: Resets the SPI2S2 block

Bit 13 **TIM11RST**: TIM11 block reset
Set and reset by software.
0: Does not reset the TIM11 block (default after reset)
1: Resets the TIM11 block

Bit 12 **TIM10RST**: TIM10 block reset
Set and reset by software.
0: Does not reset the TIM10 block (default after reset)
1: Resets the TIM10 block

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1RST**: LPTIM1 block reset
Set and reset by software.
0: Does not reset the LPTIM1 block (default after reset)
1: Resets the LPTIM1 block

Bit 8 **TIM14RST**: TIM14 block reset
Set and reset by software.
0: Does not reset the TIM14 block (default after reset)
1: Resets the TIM14 block

Bit 7 **TIM13RST**: TIM13 block reset
Set and reset by software.
0: Does not reset the TIM13 block (default after reset)
1: Resets the TIM13 block

Bit 6 **TIM12RST**: TIM12 block reset
Set and reset by software.
0: Does not reset the TIM12 block (default after reset)
1: Resets the TIM12 block

Bit 5 **TIM7RST**: TIM7 block reset
Set and reset by software.
0: Does not reset the TIM7 block (default after reset)
1: Resets the TIM7 block

Bit 4 **TIM6RST**: TIM6 block reset
Set and reset by software.
0: Does not reset the TIM6 block (default after reset)
1: Resets the TIM6 block

Bit 3 **TIM5RST**: TIM5 block reset
Set and reset by software.
0: Does not reset the TIM5 block (default after reset)
1: Resets the TIM5 block

Bit 2 **TIM4RST**: TIM4 block reset
Set and reset by software.
0: Does not reset the TIM4 block (default after reset)
1: Resets the TIM4 block

Bit 1 **TIM3RST**: TIM3 block reset
Set and reset by software.
0: Does not reset the TIM3 block (default after reset)
1: Resets the TIM3 block

Bit 0 **TIM2RST**: TIM2 block reset
Set and reset by software.
0: Does not reset the TIM2 block (default after reset)
1: Resets the TIM2 block

7.8.39 RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	UCPD1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw											

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

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **UCPD1RST**: UCPD block reset
Set and reset by software.
0: Does not reset the UCPD block (default after reset)
1: Resets the UCPD block

Bits 26:9 Reserved, must be kept at reset value.

Bit 8 **FDCANRST**: FDCAN block reset
Set and reset by software.
0: Does not reset the FDCAN block (default after reset)
1: Resets the FDCAN block

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

Bit 1 **CRSRST**: Clock recovery system reset
Set and reset by software.
0: Does not reset CRS (default after reset)
1: Resets CRS

Bit 0 Reserved, must be kept at reset value.

7.8.40 RCC APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x098

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	HRTIM 2RST	Res.	Res.	Res.	HRTIM 1RST	Res.	Res.	Res.	SPI5R ST	TIM9R ST	TIM17 RST	TIM16 RST	TIM15 RST
			rw				rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SPI4R ST	SPI1R ST	Res.	Res.	Res.	Res.	USART 10RST	USART 9RST	USART 6RST	USART 1RST	Res.	TIM20 RST	TIM8R ST	TIM1R ST
		rw	rw					rw	rw	rw	rw		rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **HRTIM2RST**: HRTIM2 block reset
Set and reset by software.
0: Does not reset the HRTIM2 block (default after reset)
1: Resets the HRTIM2 block

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **HRTIM1RST**: HRTIM1 block reset
Set and reset by software.
0: Does not reset the HRTIM1 block (default after reset)
1: Resets the HRTIM1 block

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **SPI5RST**: SPI5 block reset
Set and reset by software.
0: Does not reset the SPI5 block (default after reset)
1: Resets the SPI5 block

Bit 19 **TIM9RST**: TIM9 block reset
Set and reset by software.
0: Does not reset the TIM9 block (default after reset)
1: Resets the TIM9 block

Bit 18 **TIM17RST**: TIM17 block reset
Set and reset by software.
0: Does not reset the TIM17 block (default after reset)
1: Resets the TIM17 block

Bit 17 **TIM16RST**: TIM16 block reset
Set and reset by software.
0: Does not reset the TIM16 block (default after reset)
1: Resets the TIM16 block

Bit 16 **TIM15RST**: TIM15 block reset
Set and reset by software.
0: Does not reset the TIM15 block (default after reset)
1: Resets the TIM15 block

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

Bit 13 **SPI4RST**: SPI4 block reset
Set and reset by software.
0: Does not reset the SPI4 block (default after reset)
1: Resets the SPI4 block

Bit 12 **SPI1RST**: SPI2S1 block reset
Set and reset by software.
0: Does not reset the SPI2S1 block (default after reset)
1: Resets the SPI2S1 block

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **USART10RST**: USART10 block reset
Set and reset by software.
0: Does not reset the USART10 block (default after reset)
1: Resets the USART10 block

Bit 6 **USART9RST**: USART9 block reset
Set and reset by software.
0: Does not reset the USART9 block (default after reset)
1: Resets the USART9 block

Bit 5 **USART6RST**: USART6 block reset
Set and reset by software.
0: Does not reset the USART6 block (default after reset)
1: Resets the USART6 block

Bit 4 **USART1RST**: USART1 block reset
Set and reset by software.
0: Does not reset the USART1 block (default after reset)
1: Resets the USART1 block

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TIM20RST**: TIM20 block reset
Set and reset by software.
0: Does not reset the TIM20 block (default after reset)
1: Resets the TIM20 block

Bit 1 **TIM8RST**: TIM8 block reset
Set and reset by software.
0: Does not reset the TIM8 block (default after reset)
1: Resets the TIM8 block

Bit 0 **TIM1RST**: TIM1 block reset
Set and reset by software.
0: Does not reset the TIM1 block (default after reset)
1: Resets the TIM1 block

7.8.41 RCC APB4 peripheral reset register (RCC_APB4RSTR)

Address offset: 0x09C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP 78RST	COMP 56RST	COMP 34RST	COMP 12RST	Res.	DTSRS T	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw		rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREFR ST	Res.	Res.	LPTIM5 RST	LPTIM4 RST	LPTIM3 RST	LPTIM2 RST	Res.	Res.	Res.	SPI6R ST	Res.	LPUAR T1RST	Res.	SBSRS T	Res.
rw			rw	rw	rw	rw				rw		rw		rw	

Bit 31 **COMP78RST**: COMP78 block reset
Set and reset by software.
0: Does not reset the COMP78 block (default after reset)
1: Resets the COMP78 block

Bit 30 **COMP56RST**: COMP56 block reset
Set and reset by software.
0: Does not reset the COMP56 block (default after reset)
1: Resets the COMP56 block

Bit 29 **COMP34RST**: COMP34 block reset
Set and reset by software.
0: Does not reset the COMP34 block (default after reset)
1: Resets the COMP34 block

Bit 28 **COMP12RST**: COMP512 block reset
Set and reset by software.
0: Does not reset the COMP12 block (default after reset)
1: Resets the COMP12 block

Bit 27 Reserved, must be kept at reset value.

Bit 26 **DTSRST**: DTS block reset
Set and reset by software.
0: Does not reset the DTS block (default after reset)
1: Resets the DTS block

Bits 25:16 Reserved, must be kept at reset value.

Bit 15 **VREFRST**: VREF block reset
Set and reset by software.
0: Does not reset the VREF block (default after reset)
1: Resets the VREF block

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5RST**: LPTIM5 block reset
Set and reset by software.
0: Does not reset the LPTIM5 block (default after reset)
1: Resets the LPTIM5 block

Bit 11 **LPTIM4RST**: LPTIM4 block reset
Set and reset by software.
0: Does not reset the LPTIM4 block (default after reset)
1: Resets the LPTIM4 block

Bit 10 **LPTIM3RST**: LPTIM3 block reset
Set and reset by software.
0: Does not reset the LPTIM3 block (default after reset)
1: Resets the LPTIM3 block

Bit 9 **LPTIM2RST**: LPTIM2 block reset
Set and reset by software.
0: Does not reset the LPTIM2 block (default after reset)
1: Resets the LPTIM2 block

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **SPI6RST**: SPI/I2S6 block reset
Set and reset by software.
0: Does not reset the SPI/I2S6 block (default after reset)
1: Resets the SPI/I2S6 block

Bit 4 Reserved, must be kept at reset value.

Bit 3 **LPUART1RST**: LPUART1 block reset
Set and reset by software.
0: Does not reset the LPUART1 block (default after reset)
1: Resets the LPUART1 block

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SBSRST**: SBS block reset
Set and reset by software.
0: Does not reset the SBS block (default after reset)
1: Resets the SBS block

Bit 0 Reserved, must be kept at reset value.

7.8.42 RCC AXI clocks gating disable register (RCC_CKGDISR)

Address offset: 0x0B0

Reset value: 0x8000 0000

The dynamic power consumption can be optimized by enabling the functional clock gating.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JTAGCKG	EXTICKG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLASHCKG	AXISRAM1CKG	AXISRAM2CKG	AXISRAM3CKG	AXISRAM4CKG
rw	rw										rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XSPI1CKG	FMCCKG	AHBICCKG	AHBSCKG	Res.	Res.	Res.	Res.	Res.	NPU1CKG	NPU0CKG	CPUCKG	HPDMA1CKG	SDMMC1CKG	AHBMCKG	AXICKG
rw	rw	rw	rw						rw	rw	rw	rw	rw	rw	rw

Bit 31 **JTAGCKG**: JTAG automatic clock gating disabling
This bit is set and reset by software.
0: The clock gating is enabled. The clock is disabled except if a JTAG connection has been detected
1: The clock gating is disabled. The clock is always enabled. (default after reset)

Bit 30 **EXTICKG**: EXTI clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock is enabled after an event detection and stopped again when the event flag is cleared. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bits 29:21 Reserved, must be kept at reset value.

Bit 20 **FLASHCKG**: Flash interface (FLIFT) AXI manager port clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI manager port connected to the FLASH is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 19 **AXISRAM1CKG**: AXISRAM1 AXI manager port clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI manager port connected to the AXISRAM1 is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 18 **AXISRAM2CKG**: AXISRAM2 AXI manager port clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI manager port connected to the AXISRAM2 is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 17 **AXISRAM3CKG**: AXISRAM3 AXI manager port clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI manager port connected to the AXISRAM3 is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 16 **AXISRAM4CKG**: AXISRAM4 AXI manager port clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI manager port connected to the AXISRAM4 is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 15 **XSPI1CKG**: XSPI1 AXI manager port clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI manager port connected to the XSPI1 is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 14 **FMCCKG**: FMC AXI manager port clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI manager port connected to the FMC is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 13 **AHBICCKG**: AHB interconnect AXI manager port clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI manager port connected to the AHB interconnect is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled

Bit 12 **AHBSCKG**: AHBS AXI manager port clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI manager port connected to the CPU AHBS bus is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled

Bits 11:7 Reserved, must be kept at reset value.

Bit 6 **NPU1CKG**: AXI NPU port 1 clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI master port connected to the NPU is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 5 **NPU0CKG**: AXI NPU port 0 clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI master port connected to the NPU is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 4 **CPUCKG**: AXI CPU port clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI master port connected to the CPU is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 3 **HPDMA1CKG**: AXI HPDMA1 port clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI master port connected to the HPDMA1 is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled

Bit 2 **SDMMC1CKG**: AXI SDMMC1 port clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI master port connected to SDMMC1 is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 1 **AHBMCKG**: AXI AHB matrix port clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI master port connected to the AHB interconnect is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 0 **AXICKG**: AXI interconnect matrix clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The AXI interconnect matrix clock is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled

7.8.43 RCC Reset status register (RCC_RSR)

Address offset: 0x130

Reset value: 0x00E0 0000

Reset by power-on reset only.

Access: 0 ≤ wait state ≤ 7, word, half-word and byte access. Wait states are inserted in case of successive accesses to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LPWRRSTF	Res.	WWDGRSTF	Res.	IWDGRSTF	Res.	SFTRSTF	PORRSTF	PINRSTF	BORRSTF	Res.	Res.	Res.	OBLRSTF	RMVF
	r		r		r		r	r	r	r				r	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 Reserved, must be kept at reset value.

Bit 30 **LPWRRSTF**: reset due to illegal Stop or Standby flag (1)
Reset by software by writing the RMVF bit.
Set by hardware when the CPU goes erroneously in Stop or Standby mode,
0: No illegal reset occurred (default after power-on reset)
1: Illegal Stop or Standby reset occurred

Bit 29 Reserved, must be kept at reset value.

Bit 28 **WWDGRSTF**: window watchdog reset flag (1)
Reset by software by writing the RMVF bit.
Set by hardware when a window watchdog reset occurs.
0: No window watchdog reset occurred from WWDG (default after power-on reset)
1: Window watchdog reset occurred from WWDG

Bit 27 Reserved, must be kept at reset value.

Bit 26 **IWDGRSTF**: independent watchdog reset flag (1)
Reset by software by writing the RMVF bit.
Set by hardware when an independent watchdog reset occurs.
0: No independent watchdog reset occurred (default after power-on reset)
1: Independent watchdog reset occurred

Bit 25 Reserved, must be kept at reset value.

Bit 24 **SFTRSTF**: system reset from CPU reset flag (1)
Reset by software by writing the RMVF bit.
Set by hardware when the system reset is due to CPU.The CPU can generate a system reset by writing SYSRESETREQ bit of AIRCR register of the core M7.
0: No CPU software reset occurred (default after power-on reset)
1: A system reset has been generated by the CPU

Bit 23 **PORRSTF**: POR/PDR reset flag (1)
Reset by software by writing the RMVF bit.
Set by hardware when a POR/PDR reset occurs.
0: No POR/PDR reset occurred
1: POR/PDR reset occurred (default after power-on reset)

Bit 22 **PINRSTF**: pin reset flag (NRST) (1)
Reset by software by writing the RMVF bit.
Set by hardware when a reset from pin occurs.
0: No reset from pin occurred
1: Reset from pin occurred (default after power-on reset)

Bit 21 **BORRSTF**: BOR reset flag (1)
Reset by software by writing the RMVF bit.
Set by hardware when a BOR reset occurs (**pwr_bor_rst**).
0: No BOR reset occurred
1: BOR reset occurred (default after power-on reset)

Bits 20:18 Reserved, must be kept at reset value.

Bit 17 **OBLRSTF**: Option byte loading reset flag (1)
Reset by software by the RMVF bit.
Set by hardware when a reset from the option byte loading occurs.
0: No reset from option byte loading occurred
1: Reset from option byte loading occurred

Bit 16 **RMVF**: remove reset flags
Set by software to reset the value of the reset flags.
This bit is reset to 0 by hardware.
0: No effect (default after power-on reset)
1: Resets the value of the reset flags

Bits 15:0 Reserved, must be kept at reset value.

1. Refer to *Table 52: Reset source identification (RCC_RSR)* for details on flag behavior.

7.8.44 RCC AHB1 clock enable register (RCC_AHB1ENR)

Address offset: 0x138

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADF1EN	Res.	Res.	Res.	OTGFSEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw				rw											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA1EN	HSP1EN	Res.	Res.	Res.
											rw	rw			

Bit 31 **ADF1EN**: ADF clocks enable
Set and reset by software.
0: ADF clocks disabled (default after reset)
1: ADF clocks enabled

Bits 30:28 Reserved, must be kept at reset value.

Bit 27 **OTGFSEN**: OTGFS peripheral clocks enable
Set and reset by software.
0: OTGFS peripheral clocks disabled (default after reset)
1: OTGFS peripheral clocks enabled

Bits 26:5 Reserved, must be kept at reset value.

Bit 4 **GPDMA1EN**: GPDMA1 clock enable
Set and reset by software.
0: GPDMA1 clock disabled (default after reset)
1: GPDMA1 clock enabled

Bit 3 **HSP1EN**: HSP clock enable
Set and reset by software.
0: HSP clock disabled (default after reset)
1: HSP clock and AHBSRAM3 clock enabled

Bits 2:0 Reserved, must be kept at reset value.

7.8.45 RCC AHB2 clock enable register (RCC_AHB2ENR)

Address offset: 0x13C

Reset value: 0xE000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHBSRAM3EN	AHBSRAM2EN	AHBSRAM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMAC1EN	CORDICEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bit 31 **AHBSRAM3EN**: AHBSRAM3 clock enable
Set and reset by software.
0: AHBSRAM3 clock disabled (if HSP1EN = 0)
1: AHBSRAM3 clock enabled (default after reset)

Bit 30 **AHBSRAM2EN**: AHBSRAM2 clock enable
Set and reset by software.
0: AHBSRAM2 clock disabled
1: AHBSRAM2 clock enabled (default after reset)

Bit 29 **AHBSRAM1EN**: AHBSRAM1 clock enable
Set and reset by software.
0: AHBSRAM1 clock disabled
1: AHBSRAM1 clock enabled (default after reset)

Bits 28:16 Reserved, must be kept at reset value.

Bit 15 **FMAC1EN**: FMAC1 clock enable
Set and reset by software.
0: FMAC1 clock disabled (default after reset)
1: FMAC1 clock enabled

Bit 14 **CORDICEN**: CORDIC clock enable
Set and reset by software.
0: CORDIC clock disabled (default after reset)
1: CORDIC clock enabled

Bits 13:0 Reserved, must be kept at reset value.

7.8.46 RCC AHB3 clock enable register (RCC_AHB3ENR)

Address offset: 0x158

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DAC4EN	DAC3EN	DAC2EN	DAC1EN	Res.	Res.	ADC34EN	ADC12EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw			rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HASHEN	RNGEN
														rw	rw

Bit 31 **DAC4EN**: DAC4 peripheral clock enable
Set and reset by software.
0: DAC4 peripheral clock disabled (default after reset)
1: DAC4 peripheral clock enabled

Bit 30 **DAC3EN**: DAC3 peripheral clock enable
Set and reset by software.
0: DAC3 peripheral clock disabled (default after reset)
1: DAC3 peripheral clock enabled

Bit 29 **DAC2EN**: DAC2 peripheral clock enable
Set and reset by software.
0: DAC2 peripheral clock disabled (default after reset)
1: DAC2 peripheral clock enabled

Bit 28 **DAC1EN**: DAC1 peripheral clock enable
Set and reset by software.
0: DAC1 peripheral clock disabled (default after reset)
1: DAC1 peripheral clock enabled

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **ADC34EN**: ADC34 peripheral clock enable
Set and reset by software.
0: ADC34 peripheral clock disabled (default after reset)
1: ADC34 peripheral clock enabled

Bit 24 **ADC12EN**: ADC12 peripheral clock enable
Set and reset by software.
0: ADC12 peripheral clock disabled (default after reset)
1: ADC12 peripheral clock enabled

Bits 23:2 Reserved, must be kept at reset value.

Bit 1 **HASHEN**: HASH peripheral clock enable
Set and reset by software.
0: HASH peripheral clock disabled (default after reset)
1: HASH peripheral clock enabled

Bit 0 **RNGEN**: RNG peripheral clocks enable
Set and reset by software.
0: RNG peripheral clocks disabled (default after reset)
1: RNG peripheral clocks enabled

7.8.47 RCC AHB4 clock enable register (RCC_AHB4ENR)

Address offset: 0x140

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKPRAMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	Res.	Res.
			rw									rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKPRAMEN**: Backup RAM clock enable
Set and reset by software.
0: Backup RAM clock disabled (default after reset)
1: Backup RAM clock enabled

Bits 27:20 Reserved, must be kept at reset value.

Bit 19 **CRCEN**: CRC clock enable
Set and reset by software.
0: CRC clock disabled (default after reset)
1: CRC clock enabled

Bits 18:9 Reserved, must be kept at reset value.

Bit 8 **GPIOIEN**: GPIOI peripheral clock enable
Set and reset by software.
0: GPIOI peripheral clock disabled (default after reset)
1: GPIOI peripheral clock enabled

Bit 7 **GPIOHEN**: GPIOH peripheral clock enable
Set and reset by software.
0: GPIOH peripheral clock disabled (default after reset)
1: GPIOH peripheral clock enabled

Bit 6 **GPIOGEN**: GPIOG peripheral clock enable
Set and reset by software.
0: GPIOG peripheral clock disabled (default after reset)
1: GPIOG peripheral clock enabled

Bit 5 **GPIOFEN**: GPIOF peripheral clock enable
Set and reset by software.
0: GPIOF peripheral clock disabled (default after reset)
1: GPIOF peripheral clock enabled

Bit 4 **GPIOEEN**: GPIOE peripheral clock enable
Set and reset by software.
0: GPIOE peripheral clock disabled (default after reset)
1: GPIOE peripheral clock enabled

Bit 3 **GPIODEN**: GPIOD peripheral clock enable
Set and reset by software.
0: GPIOD peripheral clock disabled (default after reset)
1: GPIOD peripheral clock enabled

Bit 2 **GPIOCEN**: GPIOC peripheral clock enable
Set and reset by software.
0: GPIOC peripheral clock disabled (default after reset)
1: GPIOC peripheral clock enabled

Bit 1 **GPIOBEN**: GPIOB peripheral clock enable
Set and reset by software.
0: GPIOB peripheral clock disabled (default after reset)
1: GPIOB peripheral clock enabled

Bit 0 **GPIOAEN**: GPIOA peripheral clock enable
Set and reset by software.
0: GPIOA peripheral clock disabled (default after reset)
1: GPIOA peripheral clock enabled

7.8.48 RCC AHB5 clock enable register (RCC_AHB5ENR)

Address offset: 0x134

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPUEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1EN	Res.	Res.	XSPI1EN	FMCEN	Res.	Res.	Res.	HPDMA1EN
							rw			rw	rw				rw

Bit 31 **NPUEN**: NPU clock enable
Set and reset by software.
0: NPU clock disabled (default after reset)
1: NPU clock enabled

Bits 30:9 Reserved, must be kept at reset value.

Bit 8 **SDMMC1EN**: SDMMC1 and DB_SDMMC1 peripheral clocks enable
Set and reset by software.
0: SDMMC1 and DB_SDMMC1 peripheral clocks disabled (default after reset)
1: SDMMC1 and DB_SDMMC1 peripheral clocks enabled

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **XSPI1EN**: XSPI1 peripheral clocks enable
Set and reset by software.
The hardware prevents writing this bit if XSPICKP = 1.
0: XSPI1 peripheral clocks disabled (default after reset)
1: XSPI1 peripheral clocks enabled

Bit 4 **FMCEN**: FMC peripheral clocks enable
Set and reset by software.
The hardware prevents writing this bit if FMCCKP = 1.
0: FMC peripheral clocks disabled (default after reset)
1: FMC peripheral clocks enabled
The peripheral clocks of the FMC are the kernel clock selected by FMCSEL, and the **hclk5** bus interface clock.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **HPDMA1EN**: HPDMA1 peripheral clock enable
Set and reset by software.
0: HPDMA1 peripheral clock disabled (default after reset)
1: HPDMA1 peripheral clock enabled

7.8.49 RCC APB1 clock enable register 1 (RCC_APB1ENR1)

Address offset: 0x148

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8EN	UART7EN	Res.	Res.	Res.	Res.	Res.	I3C1EN	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.
rw	rw						rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3EN	SPI2EN	TIM11EN	TIM10EN	WWDGEN	Res.	LPTIM1EN	TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
rw	rw	rw	rw	rs		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UART8EN**: UART8 peripheral clocks enable
Set and reset by software.
0: UART8 peripheral clocks disabled (default after reset)
1: UART8 peripheral clocks enabled

Bit 30 **UART7EN**: UART7 peripheral clocks enable
Set and reset by software.
0: UART7 peripheral clocks disabled (default after reset)
1: UART7 peripheral clocks enabled

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **I3C1EN**: I3C1 peripheral clocks enable
Set and reset by software.
0: I3C1 peripheral clocks disabled (default after reset)
1: I3C1 peripheral clocks enabled

Bit 23 **I2C3EN**: I2C3 peripheral clocks enable
Set and reset by software.
0: I2C3 peripheral clocks disabled (default after reset)
1: I2C3 peripheral clocks enabled

Bit 22 **I2C2EN**: I2C2 peripheral clocks enable
Set and reset by software.
0: I2C2 peripheral clocks disabled (default after reset)
1: I2C2 peripheral clocks enabled

Bit 21 **I2C1EN**: I2C1 peripheral clocks enable
Set and reset by software.
0: I2C1 peripheral clocks disabled (default after reset)
1: I2C1 peripheral clocks enabled

Bit 20 **UART5EN**: UART5 peripheral clocks enable
Set and reset by software.
0: UART5 peripheral clocks disabled (default after reset)
1: UART5 peripheral clocks enabled

Bit 19 **UART4EN**: UART4 peripheral clocks enable
Set and reset by software.
0: UART4 peripheral clocks disabled (default after reset)
1: UART4 peripheral clocks enabled
The peripheral clocks of the UART4 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the **rcc_pclk1** bus interface clock.

Bit 18 **USART3EN**: USART3 peripheral clocks enable
Set and reset by software.
0: USART3 peripheral clocks disabled (default after reset)
1: USART3 peripheral clocks enabled
The peripheral clocks of the USART3 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the **rcc_pclk1** bus interface clock.

Bit 17 **USART2EN**: USART2peripheral clocks enable
Set and reset by software.
0: USART2 peripheral clocks disabled (default after reset)
1: USART2 peripheral clocks enabled
The peripheral clocks of the USART2 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the **rcc_pclk1** bus interface clock.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3EN**: SPI3 peripheral clocks enable
Set and reset by software.
0: SPI3 peripheral clocks disabled (default after reset)
1: SPI3 peripheral clocks enabled
The peripheral clocks of the SPI3 are the kernel clock selected by I2S123SRC and provided to **com_clk** input, and the **rcc_pclk1** bus interface clock.

Bit 14 **SPI2EN**: SPI2 peripheral clocks enable
Set and reset by software.
0: SPI2 peripheral clocks disabled (default after reset)
1: SPI2 peripheral clocks enabled
The peripheral clocks of the SPI2 are the kernel clock selected by I2S123SRC and provided to **com_clk** input, and the **rcc_pclk1** bus interface clock.

Bit 13 **TIM11EN**: TIM11 peripheral clock enable
Set and reset by software.
0: TIM11 peripheral clock disabled (default after reset)
1: TIM11 peripheral clock enabled

Bit 12 **TIM10EN**: TIM10 peripheral clock enable
Set and reset by software.
0: TIM10 peripheral clock disabled (default after reset)
1: TIM10 peripheral clock enabled

Bit 11 **WWDGEN**: WWDG clock enable
Set by software, and reset by hardware when a system reset occurs.
0: WWDG peripheral clock disable (default after reset)
1: WWDG peripheral clock enabled

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1EN**: LPTIM1 peripheral clocks enable
Set and reset by software.
0: LPTIM1 peripheral clocks disabled (default after reset)
1: LPTIM1 peripheral clocks enabled
The peripheral clocks of the LPTIM1 are the kernel clock selected by LPTIM1SEL and provided to **clk_lpt** input, and the **rcc_pclk1** bus interface clock.

Bit 8 **TIM14EN**: TIM14 peripheral clock enable
Set and reset by software.
0: TIM14 peripheral clock disabled (default after reset)
1: TIM14 peripheral clock enabled

Bit 7 **TIM13EN**: TIM13 peripheral clock enable
Set and reset by software.
0: TIM13 peripheral clock disabled (default after reset)
1: TIM13 peripheral clock enabled

Bit 6 **TIM12EN**: TIM12 peripheral clock enable
Set and reset by software.
0: TIM12 peripheral clock disabled (default after reset)
1: TIM12 peripheral clock enabled

Bit 5 **TIM7EN**: TIM7 peripheral clock enable
Set and reset by software.
0: TIM7 peripheral clock disabled (default after reset)
1: TIM7 peripheral clock enabled

Bit 4 **TIM6EN**: TIM6 peripheral clock enable
Set and reset by software.
0: TIM6 peripheral clock disabled (default after reset)
1: TIM6 peripheral clock enabled

Bit 3 **TIM5EN**: TIM5 peripheral clock enable
Set and reset by software.
0: TIM5 peripheral clock disabled (default after reset)
1: TIM5 peripheral clock enabled

Bit 2 **TIM4EN**: TIM4 peripheral clock enable
Set and reset by software.
0: TIM4 peripheral clock disable (default after reset)
1: TIM4 peripheral clock enabled

Bit 1 **TIM3EN**: TIM3 peripheral clock enable
Set and reset by software.
0: TIM3 peripheral clock disabled (default after reset)
1: TIM3 peripheral clock enabled

Bit 0 **TIM2EN**: TIM2 peripheral clock enable
Set and reset by software.
0: TIM2 peripheral clock disabled (default after reset)
1: TIM2 peripheral clock enabled

7.8.50 RCC APB1 clock enable register 2 (RCC_APB1ENR2)

Address offset: 0x14C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	UCPD1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	Res.	Res.	Res.	Res.	CRSEN	Res.
							rw							rw	

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **UCPD1EN**: UCPD peripheral clock enable
Set and reset by software.
0: UCPD peripheral clock disabled (default after reset)
1: UCPD peripheral clock enabled

Bits 26:9 Reserved, must be kept at reset value.

Bit 8 **FDCANEN**: FDCAN peripheral clock enable
Set and reset by software.
0: FDCAN peripheral clock disabled (default after reset)
1: FDCAN peripheral clock enabled

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

Bit 1 **CRSEN**: Clock recovery system peripheral clock enable
Set and reset by software.
0: CRS peripheral clock disabled (default after reset)
1: CRS peripheral clock enabled

Bit 0 Reserved, must be kept at reset value.

7.8.51 RCC APB2 clock enable register (RCC_APB2ENR)

Address offset: 0x150

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	HRTIM 2EN	Res.	Res.	Res.	HRTIM 1EN	Res.	Res.	Res.	SPI5E N	TIM9E N	TIM17E N	TIM16E N	TIM15E N
			rw				rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SPI4E N	SPI1E N	Res.	Res.	Res.	Res.	USART 10EN	USART 9EN	USART 6EN	USART 1EN	Res.	TIM20E N	TIM8E N	TIM1E N
		rw	rw					rw	rw	rw	rw		rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **HRTIM2EN**: HRTIM2 peripheral clock enable
Set and reset by software.
0: HRTIM2 peripheral clock disabled (default after reset)
1: HRTIM2 peripheral clock enabled

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **HRTIM1EN**: HRTIM1 peripheral clock enable
Set and reset by software.
0: HRTIM1 peripheral clock disabled (default after reset)
1: HRTIM1 peripheral clock enabled

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **SPI5EN**: SPI5 peripheral clocks enable
Set and reset by software.
0: SPI5 peripheral clocks disabled (default after reset)
1: SPI5 peripheral clocks enabled:
The peripheral clocks of the SPI5 are the kernel clock selected by SPI45SEL, and the **pclk2** bus interface clock.

Bit 19 **TIM9EN**: TIM9 peripheral clock enable
Set and reset by software.
0: TIM9 peripheral clock disabled (default after reset)
1: TIM9 peripheral clock enabled

Bit 18 **TIM17EN**: TIM17 peripheral clock enable
Set and reset by software.
0: TIM17 peripheral clock disabled (default after reset)
1: TIM17 peripheral clock enabled

Bit 17 **TIM16EN**: TIM16 peripheral clock enable
Set and reset by software.
0: TIM16 peripheral clock disabled (default after reset)
1: TIM16 peripheral clock enabled

Bit 16 **TIM15EN**: TIM15 peripheral clock enable
Set and reset by software.
0: TIM15 peripheral clock disabled (default after reset)
1: TIM15 peripheral clock enabled

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

Bit 13 **SPI4EN**: SPI4 Peripheral Clocks Enable

Set and reset by software.

0: SPI4 peripheral clocks disabled (default after reset)

1: SPI4 peripheral clocks enabled:

The peripheral clocks of the SPI4 are: the kernel clock selected by SPI45SEL, and the **pclk2** bus interface clock.

Bit 12 **SPI1EN**: SPI2S1 Peripheral Clocks Enable

Set and reset by software.

0: SPI2S1 peripheral clocks disabled (default after reset)

1: SPI2S1 peripheral clocks enabled:

The peripheral clocks of the SPI2S1 are: the kernel clock selected by SPI1SEL, and the **pclk2** bus interface clock.

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **USART10EN**: USART10 peripheral clocks enable

Set and reset by software.

0: USART10 peripheral clocks disabled (default after reset)

1: USART10 peripheral clocks enabled:

The peripheral clocks of the USART10 are the kernel clock selected by USART16910SEL, and the **pclk2** bus interface clock.

Bit 6 **USART9EN**: USART9 peripheral clocks enable

Set and reset by software.

0: USART9 peripheral clocks disabled (default after reset)

1: USART9 peripheral clocks enabled:

The peripheral clocks of the USART9 are the kernel clock selected by USART16910SEL, and the **pclk2** bus interface clock.

Bit 5 **USART6EN**: USART6 peripheral clocks enable

Set and reset by software.

0: USART6 peripheral clocks disabled (default after reset)

1: USART6 peripheral clocks enabled:

The peripheral clocks of the USART6 are the kernel clock selected by USART16910SEL, and the **pclk2** bus interface clock.

Bit 4 **USART1EN**: USART1 peripheral clocks enable

Set and reset by software.

0: USART1 peripheral clocks disabled (default after reset)

1: USART1 peripheral clocks enabled:

The peripheral clocks of the USART1 are the kernel clock selected by USART16910SEL, and the **pclk2** bus interface clock.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TIM20EN**: TIM20 peripheral clock enable
Set and reset by software.
0: TIM20 peripheral clock disabled (default after reset)
1: TIM20 peripheral clock enabled

Bit 1 **TIM8EN**: TIM8 peripheral clock enable
Set and reset by software.
0: TIM8 peripheral clock disabled (default after reset)
1: TIM8 peripheral clock enabled

Bit 0 **TIM1EN**: TIM1 peripheral clock enable
Set and reset by software.
0: TIM1 peripheral clock disabled (default after reset)
1: TIM1 peripheral clock enabled

7.8.52 RCC APB4 clock enable register (RCC_APB4ENR)

Address offset: 0x154

Reset value: 0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP78EN	COMP56EN	COMP34EN	COMP12EN	Res.	DTSEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBEN
rw	rw	rw	rw		rw										rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREFEN	Res.	Res.	LPTIM5EN	LPTIM4EN	LPTIM3EN	LPTIM2EN	Res.	Res.	Res.	SPI6EN	Res.	LPUART1EN	Res.	SBSEN	Res.
rw			rw	rw	rw	rw				rw		rw		rw	

Bit 31 **COMP78EN**: COMP78 peripheral clock enable
Set and reset by software.
0: COMP78 peripheral clock disabled (default after reset)
1: COMP78 peripheral clock enabled

Bit 30 **COMP56EN**: COMP56 peripheral clock enable
Set and reset by software.
0: COMP56 peripheral clock disabled (default after reset)
1: COMP56 peripheral clock enabled

Bit 29 **COMP34EN**: COMP34 peripheral clock enable
Set and reset by software.
0: COMP34 peripheral clock disabled (default after reset)
1: COMP34 peripheral clock enabled

Bit 28 **COMP12EN**: COMP12 peripheral clock enable
Set and reset by software.
0: COMP12 peripheral clock disabled (default after reset)
1: COMP12 peripheral clock enabled

Bit 27 Reserved, must be kept at reset value.

Bit 26 **DTSEN**: Temperature Sensor peripheral clock enable
Set and reset by software.
0: DTS peripheral clock disabled (default after reset)
1: DTS peripheral clock enabled

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

Bit 16 **RTCAPBEN**: RTC APB clock enable
Set and reset by software.
0: The register clock interface of the RTC (APB) is disabled
1: The register clock interface of the RTC (APB) is enabled (default after reset)

Bit 15 **VREFEN**: VREF peripheral clock enable
Set and reset by software.
0: VREF peripheral clock disabled (default after reset)
1: VREF peripheral clock enabled

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5EN**: LPTIM5 peripheral clocks enable
Set and reset by software.
0: LPTIM5 peripheral clocks disabled (default after reset)
1: LPTIM5 peripheral clocks enabled
The LPTIM5 kernel clock can be selected by LPTIM45SEL.

Bit 11 **LPTIM4EN**: LPTIM4 peripheral clocks enable
Set and reset by software.
0: LPTIM4 peripheral clocks disabled (default after reset)
1: LPTIM4 peripheral clocks enabled
The LPTIM4 kernel clock can be selected by LPTIM45SEL.

Bit 10 **LPTIM3EN**: LPTIM3 peripheral clocks enable
Set and reset by software.
0: LPTIM3 peripheral clocks disabled (default after reset)
1: LPTIM3 peripheral clocks enabled
The LPTIM3 kernel clock can be selected by LPTIM23SEL.

Bit 9 **LPTIM2EN**: LPTIM2 peripheral clocks enable
Set and reset by software.
0: LPTIM2 peripheral clocks disabled (default after reset)
1: LPTIM2 peripheral clocks enabled
The LPTIM2 kernel clock can be selected by LPTIM23SEL.

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **SPI6EN**: SPI/I2S6 peripheral clocks enable
Set and reset by software.
0: SPI/I2S6 peripheral clocks disabled (default after reset)
1: SPI/I2S6 peripheral clocks enabled
The peripheral clocks of the SPI/I2S6 are the kernel clock selected by SPI6SEL and provided to **com_clk** input, and the **pclk4** bus interface clock.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **LPUART1EN**: LPUART1 peripheral clocks enable
Set and reset by software.
0: LPUART1 peripheral clocks disabled (default after reset)
1: LPUART1 peripheral clocks enabled
The peripheral clocks of the LPUART1 are the kernel clock selected by LPUART1SEL and provided to UCLK input, and the **pclk4** bus interface clock.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SBSEN**: SBS peripheral clock enable
Set and reset by software.
0: SBS peripheral clock disabled (default after reset)
1: SBS peripheral clock enabled

Bit 0 Reserved, must be kept at reset value.

7.8.53 RCC AHB5 low-power clock enable register (RCC_AHB5LPENR)

Address offset: 0x15C

Reset value: 0xF800 0135

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPULPEN	ITCMLPEN	DTCM2LPEN	DTCM1LPEN	AXISRAMLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1LPEN	Res.	Res.	XSPI1LPEN	FMCLPEN	Res.	FLASHLPEN	Res.	HPDMA1LPEN
							rw			rw	rw		rw		rw

Bit 31 **NPULPEN**:
NPU low-power peripheral clock enable
Set and reset by software.
0: NPU peripheral clock disabled in low-power mode
1: NPU peripheral clock enabled in low-power mode (default after reset)

Bit 30 **ITCMLPEN**: ITCM low-power peripheral clock enable
Set and reset by software.
0: ITCM interface peripheral clock disabled in low-power mode
1: ITCM interface peripheral clock enabled in low-power mode (default after reset)

Bit 29 **DTCM2LPEN**: DTCM2 low-power peripheral clock enable
Set and reset by software.
0: DTCM2 interface peripheral clock disabled in low-power mode
1: DTCM2 interface peripheral clock enabled in low-power mode (default after reset)

Bit 28 **DTCM1LPEN**: DTCM1 low-power peripheral clock enable
Set and reset by software.
0: DTCM1 interface peripheral clock disabled in low-power mode
1: DTCM1 interface peripheral clock enabled in low-power mode (default after reset)

Bit 27 **AXISRAMLPEN**:
AXISRAM[4:1] low-power peripheral clock enable
Set and reset by software.
0: AXISRAM[4:1] interface peripheral clock disabled in low-power mode
1: AXISRAM[4:1] interface peripheral clock enabled in low-power mode (default after reset)

Bits 26:9 Reserved, must be kept at reset value.

Bit 8 **SDMMC1LPEN**: SDMMC1 and SDMMC1 delay low-power peripheral clock enable
Set and reset by software.
0: SDMMC1 and SDMMC1 delay peripheral clock disabled in low-power mode
1: SDMMC1 and SDMMC1 delay peripheral clock enabled in low-power mode (default after reset)

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **XSPI1LPEN**: XSPI1 and MCE1 low-power peripheral clock enable
Set and reset by software.
The hardware prevents writing this bit if XSPICKP = 1.
0: XSPI1 and MCE1 peripheral clock disabled in low-power mode
1: XSPI1 and MCE1 peripheral clock enabled in low-power mode (default after reset)

Bit 4 **FMCLPEN**: FMC and MCE3 peripheral clocks enable in low-power mode
Set and reset by software.
The hardware prevents writing this bit if FMCCKP = 1.
0: FMC and MCE3 peripheral clocks disabled in low-power mode
1: FMC and MCE3 peripheral clocks enabled in low-power mode (default after reset):
The peripheral clocks of the FMC are the kernel clock selected by FMCSEL, and the **hclk5** bus interface clock.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FLASHLPEN**: FLASH low-power peripheral clock enable
Set and reset by software.
0: FLASH peripheral clock disabled in low-power mode
1: FLASH peripheral clock enabled in low-power mode (default after reset)

Bit 1 Reserved, must be kept at reset value.

Bit 0 **HPDMA1LPEN**: HPDMA1 low-power peripheral clock enable
Set and reset by software.
0: HPDMA1 peripheral clock disabled in low-power mode
1: HPDMA1 peripheral clock enabled in low-power mode (default after reset)

7.8.54 RCC AHB1 low-power clock enable register (RCC_AHB1LPENR)

Address offset: 0x160

Reset value: 0x8800 0018

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADF1LPEN	Res.	Res.	Res.	OTGFSLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw				rw											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA1LPEN	HSP1LPEN	Res.	Res.	Res.
											rw	rw			

Bit 31 **ADF1LPEN**: ADF clock enable in low-power mode
Set and reset by software.
0: ADF peripheral clock disabled in low-power mode
1: ADF peripheral clock enabled in low-power mode (default after reset)

Bits 30:28 Reserved, must be kept at reset value.

Bit 27 **OTGFSLPEN**: OTGFS clock enable in low-power mode
Set and reset by software.
0: OTGFS peripheral clock disabled in low-power mode
1: OTGFS peripheral clock enabled in low-power mode (default after reset)

Bits 26:5 Reserved, must be kept at reset value.

Bit 4 **GPDMA1LPEN**: GPDMA1 clock enable in low-power mode
Set and reset by software.
0: GPDMA1 clock disabled in low-power mode
1: GPDMA1 clock enabled in low-power mode (default after reset)

Bit 3 **HSP1LPEN**: HSP clock enable in low-power mode
Set and reset by software.
0: HSP clock disabled in low-power mode
1: HSP clock and AHBSRAM3 enabled in low-power mode (default after reset)

Bits 2:0 Reserved, must be kept at reset value.

7.8.55 RCC AHB2 low-power clock enable register (RCC_AHB2LPENR)

Address offset: 0x164

Reset value: 0xE000 C000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHBSRAM3LPEN	AHBSRAM2LPEN	AHBSRAM1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMAC1LPEN	CORDICLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bit 31 **AHBSRAM3LPEN**: AHBSRAM3 clock enable in low-power mode
Set and reset by software.
0: AHBSRAM3 clock disabled in low-power mode (if HSP1LPEN = 0)
1: AHBSRAM3 clock enabled in low-power mode (default after reset)

Bit 30 **AHBSRAM2LPEN**: AHBSRAM2 clock enable in low-power mode
Set and reset by software.
0: AHBSRAM2 clock disabled in low-power mode
1: AHBSRAM2 clock enabled in low-power mode (default after reset)

Bit 29 **AHBSRAM1LPEN**: AHBSRAM1 clock enable in low-power mode
Set and reset by software.
0: AHBSRAM1 clock disabled in low-power mode
1: AHBSRAM1 clock enabled in low-power mode (default after reset)

Bits 28:16 Reserved, must be kept at reset value.

Bit 15 **FMAC1LPEN**: FMAC1 clock enable in low-power mode
Set and reset by software.
0: FMAC1 clock disabled in low-power mode
1: FMAC1 clock enabled in low-power mode (default after reset)

Bit 14 **CORDICLPEN**: CORDIC clock enable in low-power mode
Set and reset by software.
0: CORDIC clock disabled in low-power mode
1: CORDIC clock enabled in low-power mode (default after reset)

Bits 13:0 Reserved, must be kept at reset value.

7.8.56 RCC AHB3 low-power clock enable register (RCC_AHB3LPENR)

Address offset: 0x16C

Reset value: 0xF300 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DAC4LPEN	DAC3LPEN	DAC2LPEN	DAC1LPEN	Res.	Res.	ADC34LPEN	ADC12LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw			rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HASHLPEN	RNGLPEN
														rw	rw

Bit 31 **DAC4LPEN**: DAC4 peripheral clock enable in low-power mode
Set and reset by software.
0: DAC4 peripheral clock disabled in low-power mode
1: DAC4 peripheral clock enabled in low-power mode (default after reset)

Bit 30 **DAC3LPEN**: DAC3 peripheral clock enable in low-power mode
Set and reset by software.
0: DAC3 peripheral clock disabled in low-power mode
1: DAC3 peripheral clock enabled in low-power mode (default after reset)

Bit 29 **DAC2LPEN**: DAC2 peripheral clock enable in low-power mode
Set and reset by software.
0: DAC2 peripheral clock disabled in low-power mode
1: DAC2 peripheral clock enabled in low-power mode (default after reset)

Bit 28 **DAC1LPEN**: DAC1 peripheral clock enable in low-power mode
Set and reset by software.
0: DAC1 peripheral clock disabled in low-power mode
1: DAC1 peripheral clock enabled in low-power mode (default after reset)

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **ADC34LPEN**: ADC34 peripheral clock enable in low-power mode
Set and reset by software.
0: ADC34 peripheral clock disabled in low-power mode
1: ADC34 peripheral clock enabled in low-power mode (default after reset)

Bit 24 **ADC12LPEN**: ADC12 peripheral clock enable in low-power mode
Set and reset by software.
0: ADC12 peripheral clock disabled in low-power mode
1: ADC12 peripheral clock enabled in low-power mode (default after reset)

Bits 23:2 Reserved, must be kept at reset value.

Bit 1 **HASHLPEN**: HASH peripheral clock enable in low-power mode
Set and reset by software.
0: HASH peripheral clock disabled in low-power mode
1: HASH peripheral clock enabled in low-power mode (default after reset)

Bit 0 **RNGLPEN**: RNG peripheral clock enable in low-power mode
Set and reset by software.
0: RNG peripheral clocks disabled in low-power mode
1: RNG peripheral clock enabled in low-power mode (default after reset)

7.8.57 RCC AHB4 low-power clock enable register (RCC_AHB4LPENR)

Address offset: 0x168

Reset value: 0x1008 01FF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKPRAMLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCLPEN	Res.	Res.	Res.
			rw									rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKPRAMLPEN**: Backup RAM clock enable in low-power mode
Set and reset by software.
0: Backup RAM clock disabled in low-power mode
1: Backup RAM clock enabled in low-power mode (default after reset)

Bits 27:20 Reserved, must be kept at reset value.

Bit 19 **CRCLPEN**: CRC clock enable in low-power mode
Set and reset by software.
0: CRC clock disabled in low-power mode
1: CRC clock enabled in low-power mode (default after reset)

Bits 18:9 Reserved, must be kept at reset value.

Bit 8 **GPIOILPEN**: GPIOI peripheral clock enable in low-power mode
Set and reset by software.
0: GPIOI peripheral clock disabled in low-power mode
1: GPIOI peripheral clock enabled in low-power mode (default after reset)

Bit 7 **GPIOHLPEN**: GPIOH peripheral clock enable in low-power mode
Set and reset by software.
0: GPIOH peripheral clock disabled in low-power mode
1: GPIOH peripheral clock enabled in low-power mode (default after reset)

Bit 6 **GPIOGLPEN**: GPIOG peripheral clock enable in low-power mode
Set and reset by software.
0: GPIOG peripheral clock disabled in low-power mode
1: GPIOG peripheral clock enabled in low-power mode (default after reset)

Bit 5 **GPIOFLPEN**: GPIOF peripheral clock enable in low-power mode
Set and reset by software.
0: GPIOF peripheral clock disabled in low-power mode
1: GPIOF peripheral clock enabled in low-power mode (default after reset)

Bit 4 **GPIOELPEN**: GPIOE peripheral clock enable in low-power mode
Set and reset by software.
0: GPIOE peripheral clock disabled in low-power mode
1: GPIOE peripheral clock enabled in low-power mode (default after reset)

Bit 3 **GPIODLPEN**: GPIOD peripheral clock enable in low-power mode
Set and reset by software.
0: GPIOD peripheral clock disabled in low-power mode
1: GPIOD peripheral clock enabled in low-power mode (default after reset)

Bit 2 **GPIOCLPEN**: GPIOC peripheral clock enable in low-power mode
Set and reset by software.
0: GPIOC peripheral clock disabled in low-power mode
1: GPIOC peripheral clock enabled in low-power mode (default after reset)

Bit 1 **GPIOBLPEN**: GPIOB peripheral clock enable in low-power mode
Set and reset by software.
0: GPIOB peripheral clock disabled in low-power mode
1: GPIOB peripheral clock enabled in low-power mode (default after reset)

Bit 0 **GPIOALPEN**: GPIOA peripheral clock enable in low-power mode
Set and reset by software.
0: GPIOA peripheral clock disabled in low-power mode
1: GPIOA peripheral clock enabled in low-power mode (default after reset)

7.8.58 RCC APB1 low-power clock enable register 1 (RCC_APB1LPENR1)

Address offset: 0x170

Reset value: 0xC1FE FBFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 LPEN	UART7 LPEN	Res.	Res.	Res.	Res.	Res.	I3C1LP EN	I2C3LP EN	I2C2LP EN	I2C1LP EN	UART5 LPEN	UART4 LPEN	USART 3LPEN	USART 2LPEN	Res.
rw	rw						rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3LP EN	SPI2LP EN	TIM11L PEN	TIM10L PEN	WWDG LPEN	Res.	LPTIM1 LPEN	TIM14L PEN	TIM13L PEN	TIM12L PEN	TIM7LP EN	TIM6LP EN	TIM5LP EN	TIM4LP EN	TIM3LP EN	TIM2LP EN
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UART8LPEN**: UART8 peripheral clocks enable in low-power mode
Set and reset by software.
0: UART8 peripheral clocks disabled in low-power mode
1: UART8 peripheral clocks enabled in low-power mode (default after reset)

Bit 30 **UART7LPEN**: UART7 peripheral clocks enable in low-power mode
Set and reset by software.
0: UART7 peripheral clocks disabled in low-power mode
1: UART7 peripheral clocks enabled in low-power mode (default after reset)

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **I3C1LPEN**: I3C1 peripheral clocks enable in low-power mode
Set and reset by software.
0: I3C1 peripheral clocks disabled in low-power mode
1: I3C1peripheral clocks enabled in low-power mode (default after reset)

Bit 23 **I2C3LPEN**: I2C3 peripheral clocks enable in low-power mode
Set and reset by software.
0: I2C3 peripheral clocks disabled in low-power mode
1: I2C3 peripheral clocks enabled in low-power mode (default after reset)

Bit 22 **I2C2LPEN**: I2C2 peripheral clocks enable in low-power mode
Set and reset by software.
0: I2C2 peripheral clocks disabled in low-power mode
1: I2C2 peripheral clocks enabled in low-power mode (default after reset)

Bit 21 **I2C1LPEN**: I2C1 peripheral clocks enable in low-power mode
Set and reset by software.
0: I2C1 peripheral clocks disabled in low-power mode
1: I2C1 peripheral clocks enabled in low-power mode (default after reset)

Bit 20 **UART5LPEN**: UART5 peripheral clocks enable in low-power mode
Set and reset by software.
0: UART5 peripheral clocks disabled in low-power mode
1: UART5 peripheral clocks enabled in low-power mode (default after reset)

Bit 19 **UART4LPEN**: UART4 peripheral clocks enable in low-power mode
Set and reset by software.
0: UART4 peripheral clocks disabled in low-power mode
1: UART4 peripheral clocks enabled in low-power mode (default after reset)

Bit 18 **USART3LPEN**: USART3 peripheral clocks enable in low-power mode
Set and reset by software.
0: USART3 peripheral clocks disabled in low-power mode
1: USART3 peripheral clocks enabled in low-power mode (default after reset):

Bit 17 **USART2LPEN**: USART2 peripheral clocks enable in low-power mode
Set and reset by software.
0: USART2 peripheral clocks disabled in low-power mode
1: USART2 peripheral clocks enabled in low-power mode (default after reset).

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3LPEN**: SPI3 peripheral clocks enable in low-power mode
Set and reset by software.
0: SPI3 peripheral clocks disabled in low-power mode
1: SPI3 peripheral clocks enabled in low-power mode (default after reset).

Bit 14 **SPI2LPEN**: SPI2 peripheral clocks enable in low-power mode
Set and reset by software.
0: SPI2 peripheral clocks disabled in low-power mode
1: SPI2 peripheral clocks enabled in low-power mode (default after reset).

Bit 13 **TIM11LPEN**: TIM11 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM11 peripheral clock disabled in low-power mode
1: TIM11 peripheral clock enabled in low-power mode (default after reset)

Bit 12 **TIM10LPEN**: TIM10 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM10 peripheral clock disabled in low-power mode
1: TIM10 peripheral clock enabled in low-power mode (default after reset)

Bit 11 **WWDGLPEN**: WWDG clock enable in low-power mode
Set and reset by software.
0: WWDG clock disable in low-power mode
1: WWDG clock enabled in low-power mode (default after reset)

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1LPEN**: LPTIM1 peripheral clocks enable in low-power mode
Set and reset by software.
0: LPTIM1 peripheral clocks disabled in low-power mode
1: LPTIM1 peripheral clocks enabled in low-power mode (default after reset).

Bit 8 **TIM14LPEN**: TIM14 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM14 peripheral clock disabled in low-power mode
1: TIM14 peripheral clock enabled in low-power mode (default after reset)

Bit 7 **TIM13LPEN**: TIM13 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM13 peripheral clock disabled in low-power mode
1: TIM13 peripheral clock enabled in low-power mode (default after reset)

Bit 6 **TIM12LPEN**: TIM12 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM12 peripheral clock disabled in low-power mode
1: TIM12 peripheral clock enabled in low-power mode (default after reset)

Bit 5 **TIM7LPEN**: TIM7 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM7 peripheral clock disabled in low-power mode
1: TIM7 peripheral clock enabled in low-power mode (default after reset)

Bit 4 **TIM6LPEN**: TIM6 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM6 peripheral clock disabled in low-power mode
1: TIM6 peripheral clock enabled in low-power mode (default after reset)

Bit 3 **TIM5LPEN**: TIM5 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM5 peripheral clock disabled in low-power mode
1: TIM5 peripheral clock enabled in low-power mode (default after reset)

Bit 2 **TIM4LPEN**: TIM4 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM4 peripheral clock disabled in low-power mode
1: TIM4 peripheral clock enabled in low-power mode (default after reset)

Bit 1 **TIM3LPEN**: TIM3 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM3 peripheral clock disabled in low-power mode
1: TIM3 peripheral clock enabled in low-power mode (default after reset)

Bit 0 **TIM2LPEN**: TIM2 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM2 peripheral clock disabled in low-power mode
1: TIM2 peripheral clock enabled in low-power mode (default after reset)

7.8.59 RCC APB1 low-power clock enable register 2 (RCC_APB1LPENR2)

Address offset: 0x174

Reset value: 0x0800 0102

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	UCPD1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	Res.	Res.	Res.	Res.	CRSLPEN	Res.
							rw							rw	

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **UCPD1LPEN**: UCPD peripheral clock enable in low-power mode
Set and reset by software.
0: UCPD peripheral clock disabled in low-power mode
1: UCPD peripheral clock enabled in low-power mode (default after reset)

Bits 26:9 Reserved, must be kept at reset value.

Bit 8 **FDCANLPEN**: FDCAN peripheral clock enable in low-power mode
Set and reset by software.
0: FDCAN peripheral clock disabled in low-power mode
1: FDCAN peripheral clock enabled in low-power mode (default after reset)

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

Bit 1 **CRSLPEN**: Clock recovery system peripheral clock enable in low-power mode
Set and reset by software.
0: CRS peripheral clock disabled in low-power mode
1: CRS peripheral clock enabled in low-power mode (default after reset)

Bit 0 Reserved, must be kept at reset value.

7.8.60 RCC APB2 low-power clock enable register (RCC_APB2LPENR)

Address offset: 0x178

Reset value: 0x111F 30F7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	HRTIM 2LPEN	Res.	Res.	Res.	HRTIM 1LPEN	Res.	Res.	Res.	SPI5LP EN	TIM9LP EN	TIM17L PEN	TIM16L PEN	TIM15L PEN
			rw				rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SPI4LP EN	SPI1LP EN	Res.	Res.	Res.	Res.	USART 10LPE N	USART 9LPEN	USART 6LPEN	USART 1LPEN	Res.	TIM20L PEN	TIM8LP EN	TIM1LP EN
		rw	rw					rw	rw	rw	rw		rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **HRTIM2LPEN**: HRTIM2 peripheral clock enable in low-power mode
Set and reset by software.
0: HRTIM2 peripheral clock disabled in low-power mode
1: HRTIM2 peripheral clock enabled in low-power mode (default after reset)

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **HRTIM1LPEN**: HRTIM1 peripheral clock enable in low-power mode
Set and reset by software.
0: HRTIM1 peripheral clock disabled in low-power mode
1: HRTIM1 peripheral clock enabled in low-power mode (default after reset)

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **SPI5LPEN**: SPI5 peripheral clocks enable in low-power mode
Set and reset by software.
0: SPI5 peripheral clocks disabled in low-power mode
1: SPI5 peripheral clocks enabled in low-power mode (default after reset)
The peripheral clocks of the SPI5 are the kernel clock selected by SPI45SEL and provided to **com_clk** input, and the **pclk2** bus interface clock.

Bit 19 **TIM9LPEN**: TIM9 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM9 peripheral clock disabled in low-power mode
1: TIM9 peripheral clock enabled in low-power mode (default after reset)

Bit 18 **TIM17LPEN**: TIM17 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM17 peripheral clock disabled in low-power mode
1: TIM17 peripheral clock enabled in low-power mode (default after reset)

Bit 17 **TIM16LPEN**: TIM16 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM16 peripheral clock disabled in low-power mode
1: TIM16 peripheral clock enabled in low-power mode (default after reset)

Bit 16 **TIM15LPEN**: TIM15 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM15 peripheral clock disabled in low-power mode
1: TIM15 peripheral clock enabled in low-power mode (default after reset)

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

Bit 13 **SPI4LPEN**: SPI4 peripheral clock enable in low-power mode
Set and reset by software.
0: SPI4 peripheral clocks disabled in low-power mode
1: SPI4 peripheral clocks enabled in low-power mode (default after reset)
The peripheral clocks of the SPI4 are: the kernel clock selected by SPI45SEL and provided to **com_clk** input, and the **pclk2** bus interface clock.

Bit 12 **SPI1LPEN**: SPI2S1 peripheral clock enable in low-power mode
Set and reset by software.
0: SPI2S1 peripheral clocks disabled in low-power mode
1: SPI2S1 peripheral clocks enabled in low-power mode (default after reset)
The peripheral clocks of the SPI2S1 are: the kernel clock selected by I2S1SEL and provided to **spi_ker_ck** input, and the **pclk2** bus interface clock.

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **USART10LPEN**: USART10 peripheral clock enable in low-power mode
Set and reset by software.
0: USART10 peripheral clocks disabled in low-power mode
1: USART10 peripheral clocks enabled in low-power mode (default after reset)
The peripheral clocks of the USART0 are the kernel clock selected by USART16910SEL and provided to UCLK inputs, and the **pclk2** bus interface clock.

Bit 6 **USART9LPEN**: USART9 peripheral clock enable in low-power mode
Set and reset by software.
0: USART9 peripheral clocks disabled in low-power mode
1: USART9 peripheral clocks enabled in low-power mode (default after reset)
The peripheral clocks of the USART9 are the kernel clock selected by USART16910SEL and provided to UCLK inputs, and the **pclk2** bus interface clock.

Bit 5 **USART6LPEN**: USART6 peripheral clock enable in low-power mode
Set and reset by software.
0: USART6 peripheral clocks disabled in low-power mode
1: USART6 peripheral clocks enabled in low-power mode (default after reset)
The peripheral clocks of the USART6 are the kernel clock selected by USART16910SEL and provided to UCLK inputs, and the **pclk2** bus interface clock.

Bit 4 **USART1LPEN**: USART1 peripheral clock enable in low-power mode
Set and reset by software.
0: USART1 peripheral clocks disabled in low-power mode
1: USART1 peripheral clocks enabled in low-power mode (default after reset)
The peripheral clocks of the USART1 are the kernel clock selected by USART16910SEL and provided to UCLK inputs, and the **pclk2** bus interface clock.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TIM20LPEN**: TIM20 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM20 peripheral clock disabled in low-power mode
1: TIM20 peripheral clock enabled in low-power mode (default after reset)

Bit 1 **TIM8LPEN**: TIM8 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM8 peripheral clock disabled in low-power mode
1: TIM8 peripheral clock enabled in low-power mode (default after reset)

Bit 0 **TIM1LPEN**: TIM1 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM1 peripheral clock disabled in low-power mode
1: TIM1 peripheral clock enabled in low-power mode (default after reset)

7.8.61 RCC APB4 low-power clock enable register (RCC_APB4LPENR)

Address offset: 0x17C

Reset value: 0xF401 9E2A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP78LPEN	COMP56LPEN	COMP34LPEN	COMP12LPEN	Res.	DTSLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBLPEN
rw	rw	rw	rw		rw										rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREFLPEN	Res.	Res.	LPTIM5LPEN	LPTIM4LPEN	LPTIM3LPEN	LPTIM2LPEN	Res.	Res.	Res.	SPI6LPEN	Res.	LPUART1LPEN	Res.	SBSLPEN	Res.
rw			rw	rw	rw	rw				rw		rw		rw	

Bit 31 **COMP78LPEN**: temperature sensor peripheral clock enable in low-power mode
Set and reset by software.
0: COMP78 peripheral clock disabled in low-power mode
1: COMP78 peripheral clock enabled in low-power mode (default after reset)

Bit 30 **COMP56LPEN**: temperature sensor peripheral clock enable in low-power mode
Set and reset by software.
0: COMP56 peripheral clock disabled in low-power mode
1: COMP56 peripheral clock enabled in low-power mode (default after reset)

Bit 29 **COMP34LPEN**: temperature sensor peripheral clock enable in low-power mode
Set and reset by software.
0: COMP34 peripheral clock disabled in low-power mode
1: COMP34 peripheral clock enabled in low-power mode (default after reset)

Bit 28 **COMP12LPEN**: temperature sensor peripheral clock enable in low-power mode
Set and reset by software.
0: COMP12 peripheral clock disabled in low-power mode
1: COMP12 peripheral clock enabled in low-power mode (default after reset)

Bit 27 Reserved, must be kept at reset value.

Bit 26 **DTSLPEN**: temperature sensor peripheral clock enable in low-power mode
Set and reset by software.
0: DTS peripheral clock disabled in low-power mode
1: DTS peripheral clock enabled in low-power mode (default after reset)

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

Bit 16 **RTCAPBLPEN**: RTC APB clock enable in low-power mode
Set and reset by software.
0: The register clock interface of the RTC (APB) is disabled in low-power mode
1: The register clock interface of the RTC (APB) is enabled in low-power mode (default after reset)

Bit 15 **VREFLPEN**: VREF peripheral clock enable in low-power mode
Set and reset by software.
0: VREF peripheral clock disabled in low-power mode
1: VREF peripheral clock enabled in low-power mode (default after reset)

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5LPEN**: LPTIM5 peripheral clocks enable in low-power mode
Set and reset by software.
0: LPTIM5 peripheral clocks disabled in low-power mode
1: LPTIM5 peripheral clocks enabled in low-power mode (default after reset)
The peripheral clocks of the LPTIM5 are the kernel clock selected by LPTIM45SEL and provided to **clk_lpt** input, and the **pclk4** bus interface clock.

Bit 11 **LPTIM4LPEN**: LPTIM4 peripheral clocks enable in low-power mode
Set and reset by software.
0: LPTIM4 peripheral clocks disabled in low-power mode
1: LPTIM4 peripheral clocks enabled in low-power mode (default after reset)
The peripheral clocks of the LPTIM4 are the kernel clock selected by LPTIM45SEL and provided to **clk_lpt** input, and the **pclk4** bus interface clock.

Bit 10 **LPTIM3LPEN**: LPTIM3 peripheral clocks enable in low-power mode
Set and reset by software.
0: LPTIM3 peripheral clocks disabled in low-power mode
1: LPTIM3 peripheral clocks enabled in low-power mode (default after reset)
The peripheral clocks of the LPTIM3 are the kernel clock selected by LPTIM23SEL and provided to **clk_lpt** input, and the **pclk4** bus interface clock.

Bit 9 **LPTIM2LPEN**: LPTIM2 peripheral clocks enable in low-power mode
Set and reset by software.
0: LPTIM2 peripheral clocks disabled in low-power mode
1: LPTIM2 peripheral clocks enabled in low-power mode (default after reset)
The peripheral clocks of the LPTIM2 are the kernel clock selected by LPTIM23SEL and provided to **clk_lpt** input, and the **pclk4** bus interface clock.

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **SPI6LPEN**: SPI/I2S6 peripheral clocks enable in low-power mode
Set and reset by software.
0: SPI/I2S6 peripheral clocks disabled in low-power mode
1: SPI/I2S6 peripheral clocks enabled in low-power mode (default after reset)
The peripheral clocks of the SPI/I2S6 are the kernel clock selected by SPI6SEL and provided to **com_ck** input, and the **rcc_pclk4** bus interface clock.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **LPUART1LPEN**: LPUART1 peripheral clocks enable in low-power mode
Set and reset by software.
0: LPUART1 peripheral clocks disabled in low-power mode
1: LPUART1 peripheral clocks enabled in low-power mode (default after reset)
The peripheral clocks of the LPUART1 are the kernel clock selected by LPUART1SEL and provided to UCLK input, and the **rcc_pclk4** bus interface clock.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SBSLPEN**: SBS peripheral clock enable in low-power mode
Set and reset by software.
0: SBS peripheral clock disabled in low-power mode
1: SBS peripheral clock enabled in low-power mode (default after reset)

Bit 0 Reserved, must be kept at reset value.

7.9 RCC register map

Table 67. RCC register map and reset values

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RCC_CR	Res.	Res.	PLL3RDY	PLL3ON	PLL2RDY	PLL2ON	PLL1RDY	PLL1ON	Res.	Res.	Res.	HSECSSON	HSEEXT	HSEBYP	HSERDY	HSEON	Res.	Res.	HSI48RDY	HSI48ON	Res.	Res.	CSIKERON	CSIRDY	CSION	Res.	HSIDIVF	HSIDIV[1:0]		HSIRDY	HSIKERON	HSION
	Reset value			0	0	0	0	0	0				0	0	0	0	0			0	0			0	0	0		1	0	0	1	0	1
0x004	RCC_HSICFGR	Res.	HSITRIM[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSICAL[11:0]											
	Reset value		1	0	0	0	0	0	0													X	X	X	X	X	X	X	X	X	X	X	X
0x008	RCC_CRRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48CAL[9:0]									
	Reset value																							X	X	X	X	X	X	X	X	X	X
0x00C	RCC_CSICFGR	Res.	Res.	CSITRIM[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSICAL[7:0]							
	Reset value			1	0	0	0	0	0																	X	X	X	X	X	X	X	X
0x010	RCC_CFGR	MCO2SEL[2:0]		MCO2PRE[3:0]				MCO1SEL[2:0]				MCO1PRE[3:0]				Res.	Res.	Res.	Res.	RTCPRE[5:0]						STOPKERWUCK	STOPWUCK	Res.	SWS[2:0]		Res.	SW[2:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0		0	0		0	0
0x014	Reserved																																
0x018	RCC_CDCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPRE[3:0]			
	Reset value																													0	0	0	0
0x01C	RCC_BMCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPRE[3:0]			
	Reset value																													0	0	0	0
0x020	RCC_APBCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPRE4[2:0]			Res.	PPRE2[2:0]			Res.	PPRE1[2:0]		
	Reset value																						0	0	0		0	0	0		0	0	0
0x024	Reserved																																
0x028	RCC_PLLCKSELR	Res.	Res.	Res.	Res.	Res.	Res.	DIVM3[5:0]						Res.	Res.	DIVM2[5:0]						Res.	Res.	DIVM1[5:0]						Res.	Res.	PLLSRC[1:0]	
	Reset value							1	0	0	0	0	0			1	0	0	0	0	0			1	0	0	0	0	0			0	0

Table 67. RCC register map and reset values

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	RCC_PLLCFGR	Res.	PLL3SEN	PLL3REN	PLL3QEN	PLL3PEN	PLL3RGE[1:0]		PLL3SSCGEN	PLL3VCOSEL	PLL3FRACEN	Res.	PLL2TEN	PLL2SEN	PLL2REN	PLL2QEN	PLL2PEN	PLL2RGE[1:0]		PLL2SSCGEN	PLL2VCOSEL	PLL2FRACLEN	Res.	Res.	PLL1SEN	Res.	PLL1QEN	PLL1PEN	PLL1RGE[1:0]		PLL1SSCGEN	PLL1VCOSEL	PLL1FRACEN
	Reset value		0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0			0		0	0	0	0	0	0	0
0x030	RCC_PLL1DIVR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVQ[6:0]							DIVP[6:0]							DIVN[8:0]								
	Reset value										0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x034	RCC_PLL1FRACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRACN[12:0]													Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x038	RCC_PLL2DIVR1	Res.	DIVR[6:0]							Res.	DIVQ[6:0]							DIVP[6:0]							DIVN[8:0]								
	Reset value		0	0	0	0	0	0	1		0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x03C	RCC_PLL2FRACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRACN[12:0]													Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x040	RCC_PLL3DIVR1	Res.	DIVR[6:0]							Res.	DIVQ[6:0]							DIVP[6:0]							DIVN[8:0]								
	Reset value		0	0	0	0	0	0	1		0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x044	RCC_PLL3FRACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRACN[12:0]													Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x048	Reserved																																
0x04C	RCC_CCIPR1	Res.	Res.	CKPERSEL[1:0]		Res.	ADCSEL[2:0]			Res.	ADF1SEL[2:0]			ADCDIV[3:0]				OTGFSSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XSPI1SEL[1:0]		Res.	SDMMC1SEL	FMCSEL[1:0]	
	Reset value			0	0		0	0	0		0	0	0	0	0	0	0	0	0								0	0		0	0	0
0x050	RCC_CCIPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANSEL[1:0]		Res.	Res.	Res.	LPTIM1SEL[2:0]			Res.	Res.	I3C1SEL[1:0]		Res.	Res.	I2C123SEL[1:0]		Res.	SPI23SEL[2:0]			Res.	UART234578SEL[2:0]		
	Reset value									0	0				0	0	0			0	0			0	0		0	0	0		0	0	0

Table 67. RCC register map and reset values

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x054	RCC_CCIPR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI1SEL[2:0]			Res.	SPI45SEL[2:0]			Res.	USART16910SEL[2:0]		
	Reset value																						0	0	0		0	0	0		0	0	0
0x058	RCC_CCIPR4	Res.	DAC4SEL[2:0]			Res.	DAC3SEL[2:0]			Res.	DAC2SEL[2:0]			DAC1SHSEL	DAC1SEL[2:0]			Res.	LPTIM45SEL[2:0]			Res.	LPTIM23SEL[2:0]			Res.	SPI6SEL[2:0]			Res.	LPUART1SEL[2:0]		
	Reset value		0	0	0		0	0	0		0	0	0	0	0	0	0		0	0	0		0	0	0		0	0	0		0	0	0
0x05C	Reserved																																
0x060	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSECSSIE	PLL3RDYIE	PLL2RDYIE	PLL1RDYIE	HSI48RDYIE	CSIRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x064	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSF	LSECSSF	PLL3RDYF	PLL2RDYF	PLL1RDYF	HSI48RDYF	CSIRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x068	RCC_CICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSC	LSECSSC	PLL3RDYC	PLL2RDYC	PLL1RDYC	HSI48RDYC	CSIRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x06C	Reserved																																
0x070	RCC_BDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VSWRST	RTCEN	Res.	Res.	LSECSSRA	Res.	Res.	RTCSEL[1:0]		LSEEXT	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
	Reset value																0	0			0			0	0	0	0	0	1	0	0	0	0
0x074	RCC_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSIRDY	LSION
	Reset value																															0	0
0x078	Reserved																																

Table 67. RCC register map and reset values

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x07C	RCC_AHB5RSTR	NPURST																							SDMMC1RST			XSPI1RST	FMCRST				HPDMA1RST
	Reset value	0																							0			0	0				0
0x080	RCC_AHB1RSTR	ADF1RST				OTGFSRST																							GPDMA1RST	HSP1RST			
	Reset value	0				0																							0	0			
0x084	RCC_AHB2RSTR																	FMAC1RST	CORDICRST														
	Reset value																	0	0														
0x088	RCC_AHB4RSTR													CRCRST											GPIOIRST	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
	Reset value													0											0	0	0	0	0	0	0	0	0
0x08C	Reserved																																
0x090	RCC_APB1RSTR1	UART8RST	UART7RST						I3C1RST	I2C3RST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST		SPI3RST	SPI2RST	TIM11RST	TIM10RST			LPTIM1RST	TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
	Reset value	0	0						0	0	0	0	0	0	0	0		0	0	0	0			0	0	0	0	0	0	0	0	0	0
0x094	RCC_APB1RSTR2					UCPD1RST																			FDCANRST								
	Reset value					0																			0								
0x098	RCC_APB2RSTR				HRTIM2RST				HRTIM1RST				SPI5RST	TIM9RST	TIM17RST	TIM16RST	TIM15RST			SPI4RST	SPI1RST					USART10RST	USART9RST	USART6RST	USART1RST				
	Reset value				0				0				0	0	0	0	0			0	0					0	0	0	0				
0x09C	RCC_APB4RSTR	COMP78RST	COMP56RST	COMP34RST	COMP12RST		DTSRST											VREFRST			LPTIM5RST	LPTIM4RST	LPTIM3RST	LPTIM2RST				SPI6RST		LPUART1RST		SBSRST		
	Reset value	0	0	0	0		0											0			0	0	0	0				0		0		0		

Table 67. RCC register map and reset values

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0A0	Reserved																																
0x0A4	RCC_AHB3RSTR	DAC4RST	DAC3RST	DAC2RST	DAC1RST	Res.	Res.	ADC34RST	ADC12RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HASHRST	RNGRST
	Reset value	0	0	0	0			0	0																							0	0
0x0A8 - 0x0AC	Reserved																																
0x0B0	RCC_CKGDISR	JTAGCKG	EXTICKG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLASHCKG	AXISRAM1CKG	AXISRAM2CKG	AXISRAM3CKG	AXISRAM4CKG	XSPI1CKG	FMCCKG	AHBICCKG	AHBSCKG	Res.	Res.	Res.	Res.	Res.	NPU1CKG	NPU0CKG	CPUCKG	HPDMA1CKG	SDMMC1CKG	AHBMCKG	AXICKG
	Reset value	1	0										0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0
0x0B4 - 0x0BC	Reserved																																
0x0C0	RCC_PLL1DIVR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVS[2:0]		
	Reset value																							0							0	0	1
0x0C4	RCC_PLL2DIVR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVT[2:0]								DIVS[2:0]		
	Reset value																						0	0	1						0	0	1
0x0C8	RCC_PLL3DIVR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVS[2:0]		
	Reset value																							0							0	0	1
0x0CC	RCC_PLL1SSCGR	Res.	INCSTEP[14:0]															SPREADSEL	RPDFNDIS	TPDFNDIS	MODPER[12:0]												
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D0	RCC_PLL2SSCGR	Res.	INCSTEP[14:0]															SPREADSEL	RPDFNDIS	TPDFNDIS	MODPER[12:0]												
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 67. RCC register map and reset values

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D4	RCC_PLL3SSCGR	Res.	INCSTEP[14:0]															SPREADSEL	RPDFNDIS	TPDFNDIS	MODPER[12:0]												
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D8 - 0x0FC	Reserved																																
0x100	RCC_CKPROTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCSWP[2:0]		Res.	Res.	Res.	Res.	Res.	Res.	XSPI1SWP[2:0]			Res.	Res.	FMCCKP	XSPICKP
	Reset value																		0	0	0						0	0	0			0	0
0x104 - 0x12C	Reserved																																
0x130	RCC_RSR	Res.	LPWRRSTF	Res.	WWDGRSTF	Res.	IWDGRSTF	Res.	SFTRSTF	PORRSTF	PINRSTF	BORRSTF	Res.	Res.	Res.	OBLRSTF	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0		0		0		0	1	1	1				0	0																
0x134	RCC_AHB5ENR	NPUEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1EN	Res.	Res.	XSPI1EN	FMCEN	Res.	Res.	Res.	HPDMA1EN
	Reset value	0																							0			0	0				0
0x138	RCC_AHB1ENR	ADF1EN	Res.	Res.	Res.	OTGFSEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA1EN	HSP1EN	Res.	Res.	Res.
	Reset value	0				0																							0	0			
0x13C	RCC_AHB2ENR	AHBSRAM3EN	AHBSRAM2EN	AHBSRAM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMAC1EN	CORDICEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	1	1	1														0	0														
0x140	RCC_AHB4ENR	Res.	Res.	Res.	BKPRAMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value				0									0											0	0	0	0	0	0	0	0	0
0x144	Reserved																																

Table 67. RCC register map and reset values

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x148	RCC_APB1ENR1	UART8EN	UART7EN	Res.	Res.	Res.	Res.	Res.	I3C1EN	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.	SPI3EN	SPI2EN	TIM11EN	TIM10EN	WWDGEN	Res.	LPTIM1EN	TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	Reset value	0	0						0	0	0	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0x14C	RCC_APB1ENR2	Res.	Res.	Res.	Res.	UCPD1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	Res.	Res.	Res.	Res.	CRSEN	Res.
	Reset value					0																			0							0	
0x150	RCC_APB2ENR	Res.	Res.	Res.	HRTIM2EN	Res.	Res.	Res.	HRTIM1EN	Res.	Res.	Res.	SPI5EN	TIM9EN	TIM17EN	TIM16EN	TIM15EN	Res.	Res.	SPI4EN	SPI1EN	Res.	Res.	Res.	Res.	USART10EN	USART9EN	USART6EN	USART1EN	Res.	TIM20EN	TIM8EN	TIM1EN	
	Reset value				0				0				0	0	0	0	0			0	0					0	0	0	0		0	0	0	
0x154	RCC_APB4ENR	COMP78EN	COMP56EN	COMP34EN	COMP12EN	Res.	DTSEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBEN	VREFEN	Res.	Res.	LPTIM5EN	LPTIM4EN	LPTIM3EN	LPTIM2EN	Res.	Res.	Res.	SPI6EN	Res.	LPUART1EN	Res.	SBSEN	Res.	
	Reset value	0	0	0	0		0										1	0			0	0	0	0				0		0	0	0	
0x158	RCC_AHB3ENR	DAC4EN	DAC3EN	DAC2EN	DAC1EN	Res.	Res.	ADC34EN	ADC12EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HASHEN	RNGEN
	Reset value	0	0	0	0			0	0																						0	0
0x15C	RCC_AHB5LPENR	NPULPEN	ITCMLPEN	DTCM2LPEN	DTCM1LPEN	AXISRAMLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC1LPEN	Res.	Res.	XSPI1LPEN	FMCLPEN	Res.	FLASHLPEN	Res.	HPDMA1LPEN
	Reset value	1	1	1	1	1																			1			1	1		1		1
0x160	RCC_AHB1LPENR	ADF1LPEN	Res.	Res.	Res.	OTGFSLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA1LPEN	HSP1LPEN	Res.	Res.	Res.
	Reset value	1				1																							1	1			
0x164	RCC_AHB2LPENR	AHBSRAM3LPEN	AHBSRAM2LPEN	AHBSRAM1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMAC1LPEN	CORDICLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	1	1	1														1	1														

Table 67. RCC register map and reset values

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x168	RCC_AHB4LPENR	Res.	Res.	Res.	BKPRAMLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
	Reset value				1									1											1	1	1	1	1	1	1	1	1
0x16C	RCC_AHB3LPENR	DAC4LPEN	DAC3LPEN	DAC2LPEN	DAC1LPEN	Res.	Res.	ADC34LPEN	ADC12LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HASHLPEN	RNGLPEN
	Reset value	1	1	1	1			1	1																							1	1
0x170	RCC_APB1LPENR1	UART8LPEN	UART7LPEN	Res.	Res.	Res.	Res.	Res.	Res.	I2C3LPEN	I2C2LPEN	I2C1LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	Res.	SPI3LPEN	SPI2LPEN	TIM11LPEN	TIM10LPEN	WWDGLPEN	Res.	LPTIM1LPEN	TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
	Reset value	1	1							1	1	1	1	1	1	1		1	1	1	1	1		1	1	1	1	1	1	1	1	1	1
0x174	RCC_APB1LPENR2	Res.	Res.	Res.	Res.	UCPD1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	Res.	Res.	Res.	Res.	CRSLPEN	Res.
	Reset value					1																			1							1	
0x178	RCC_APB2LPENR	Res.	Res.	Res.	HRTIM2LPEN	Res.	Res.	Res.	HRTIM1LPEN	Res.	Res.	Res.	SPI5LPEN	TIM9LPEN	TIM17LPEN	TIM16LPEN	TIM15LPEN	Res.	Res.	SPI4LPEN	SPI1LPEN	Res.	Res.	Res.	Res.	USART10LPEN	USART9LPEN	USART6LPEN	USART1LPEN	Res.	TIM20LPEN	TIM8LPEN	TIM1LPEN
	Reset value				1				1				1	1	1	1	1			1	1					1	1	1	1		1	1	1
0x17C	RCC_APB4LPENR	COMP78LPEN	COMP56LPEN	COMP34LPEN	COMP12LPEN	Res.	DTSLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBLPEN	VREFLPEN	Res.	Res.	LPTIM5LPEN	LPTIM4LPEN	LPTIM3LPEN	LPTIM2LPEN	Res.	Res.	Res.	SPI6LPEN	Res.	LPUART1LPEN	Res.	SBSLPEN	Res.
	Reset value	1	1	1	1		1										1	1			1	1	1	1				1		1		1	

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

8 Clock recovery system (CRS)

8.1 CRS introduction

The clock recovery system (CRS) is an advanced digital controller acting on the internal fine-granularity trimmable RC oscillator HSI48. The CRS provides powerful means to evaluate the oscillator output frequency, based on comparison with a selectable synchronization signal. The CRS can perform automatic trimming adjustments based on the measured frequency error value, while keeping the possibility of a manual trimming.

The CRS is ideally suited to provide a precise clock to the USB peripheral. In this case, the synchronization signal can be derived from the start-of-frame (SOF) packet signalization on the USB bus, sent by a USB host at 1 ms intervals.

The synchronization signal can also be derived from alternative synchronization or generated by the user software.

8.2 CRS main features

- Selectable synchronization source with programmable prescaler and polarity (see *Section 8.4.2: CRS internal signals*)
- Possibility to generate synchronization pulses by software
- Automatic oscillator trimming capability with no need of CPU action
- Manual control option for faster startup convergence
- 16-bit frequency error counter with automatic error value capture and reload
- Programmable limit for automatic frequency error value evaluation and status reporting
- Maskable interrupts/events:
  - Expected synchronization (ESYNC)
  - Synchronization OK (SYNCOK)
  - Synchronization warning (SYNCWARN)
  - Synchronization or trimming error (ERR)

8.3 CRS implementation

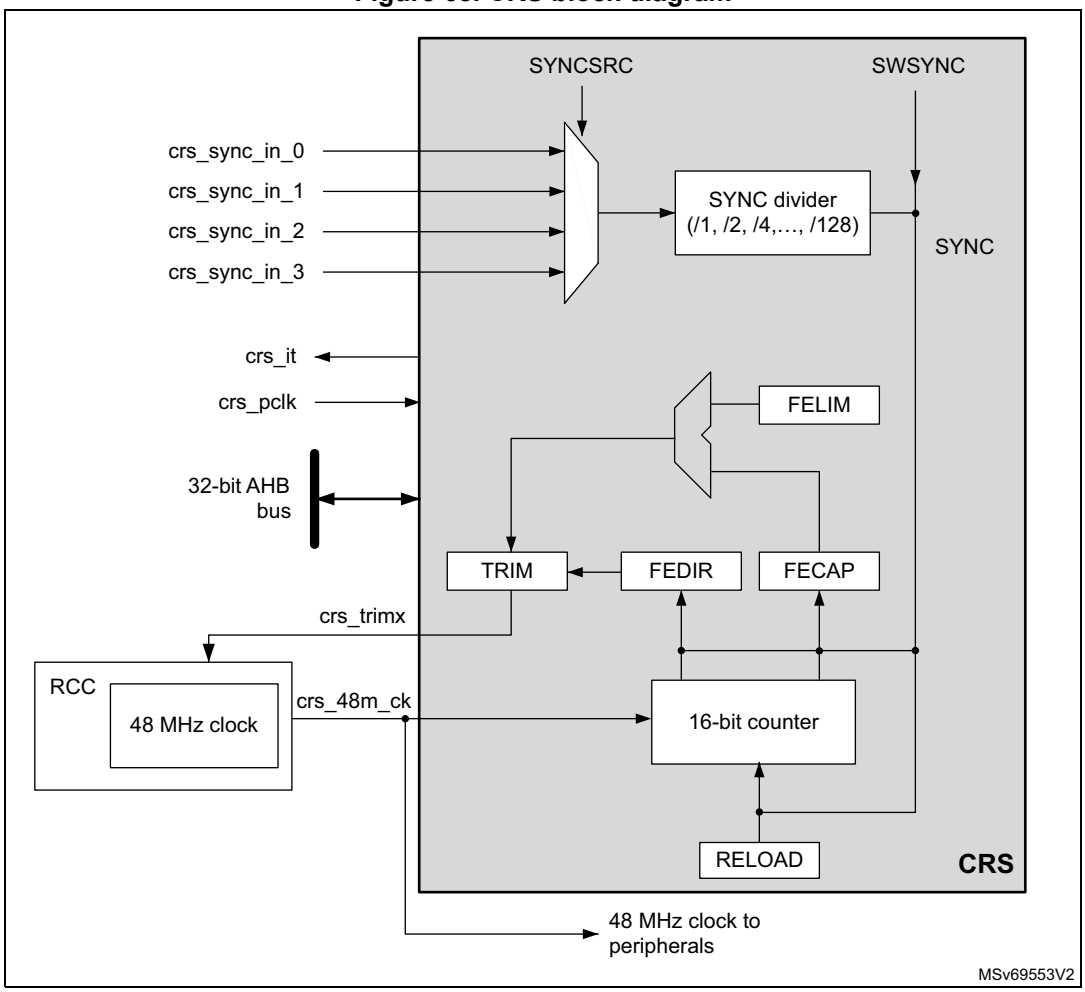
Table 68. CRS features

Feature	CRS
TRIM width	6 bits

8.4 CRS functional description

8.4.1 CRS block diagram

Figure 68. CRS block diagram



8.4.2 CRS internal signals

Below the list of CRS internal signals.

Table 69. CRS internal input/output signals

Internal signal name	Signal type	Description
crs_it	Digital output	CRS interrupt
crs_pclk	Digital input	AHB bus clock
crs_48m_ck	Digital input	HSI48 48 MHz clock
crs_trim[0:5]	Digital output	HSI48 oscillator smooth trimming value

Table 69. CRS internal input/output signals (continued)

Internal signal name	Signal type	Description
crs_sync_in_0	Digital input	SYNC signal source
crs_sync_in_1		
crs_sync_in_2		
crs_sync_in_3		

Table 70. CRS interconnection

Internal signal name	Source
crs_sync_in_0	CRS_SYNC pin selected as SYNC signal source
crs_sync_in_1	LSE selected as SYNC signal source
crs_sync_in_2	OTG FS SOF selected as SYNC signal source (default)
crs_sync_in_3	Reserved

8.4.3 Synchronization input

The CRS synchronization source (crs_sync_in_x) can be selected through SYNCSRC[1:0] bitfield of CRS_CFGR register (refer to *Section 8.4.2: CRS internal signals* for the possible sources). For a better robustness of the crs_sync_in_x input, a simple digital filter (2 out of 3 majority votes, sampled by the 48 MHz clock) is implemented to filter out glitches. In addition, the source signal has a configurable polarity (selected through SYNCPOL bit of CRS_CFGR). The signal can then be divided by a programmable binary prescaler to obtain a synchronization signal in a suitable frequency range (usually around 1 kHz).

For more information on the CRS synchronization source configuration, refer to CRS_CFGR register.

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

For more information on the CRS synchronization source configuration, refer to *CRS configuration register (CRS_CFGR)*.

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

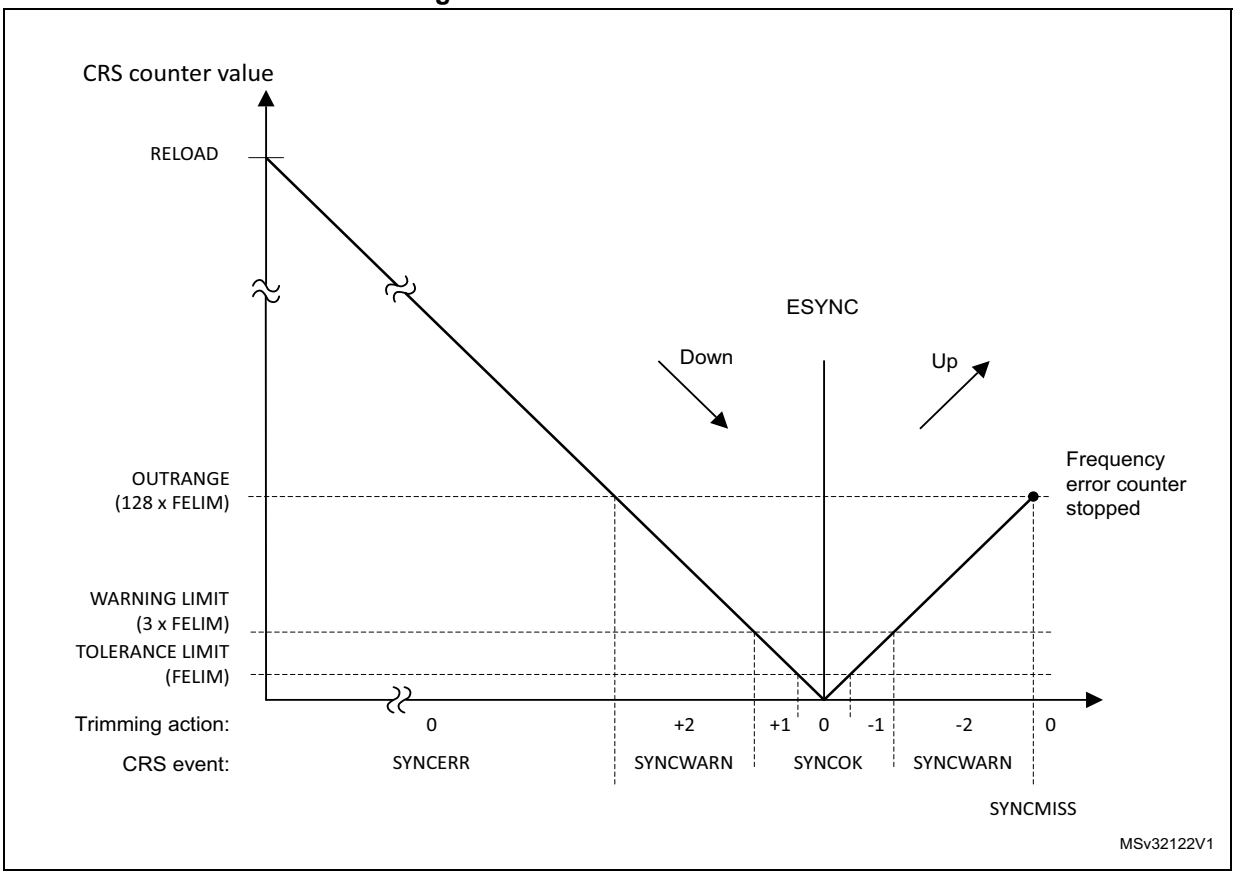
8.4.4 Frequency error measurement

The frequency error counter is a 16-bit down/up counter, reloaded with the RELOAD value on each SYNC event. It starts counting down until it reaches the zero value, where the ESYNC (expected synchronization) event is generated. Then it starts counting up to the OUTRANGE limit, where it eventually stops (if no SYNC event is received), and generates a SYNCMISS event. The OUTRANGE limit is defined as the frequency error limit (FELIM field of the CRS_CFGR register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP (frequency error capture) field and in the FEDIR (frequency error direction) bit of the CRS_ISR register. When the SYNC event is detected during the down-counting phase (before reaching the zero value), it means that the actual

frequency is lower than the target (the TRIM value must be incremented). When it is detected during the up-counting phase, it means that the actual frequency is higher (the TRIM value must be decremented).

Figure 69. CRS counter behavior



8.4.5 Frequency error evaluation and automatic trimming

The measured frequency error is evaluated by comparing its value with a set of limits:

- TOLERANCE LIMIT, given directly in the FELIM field of the CRS_CFGR register
- WARNING LIMIT, defined as 3 × FELIM value
- OUTRANGE (error limit), defined as 128 × FELIM value

The result of this comparison is used to generate the status indication and also to control the automatic trimming, which is enabled by setting the AUTOTRIMEN bit in the CRS_CR register:

- When the frequency error is below the tolerance limit, it means that the actual trimming value in the TRIM field is the optimal one, hence no trimming action is needed.
  - SYNCOK status indicated
  - TRIM value not changed in AUTOTRIM mode
- When the frequency error is below the warning limit but above or equal to the tolerance limit, it means that some trimming action is necessary but that adjustment by one trimming step is enough to reach the optimal TRIM value.
  - SYNCOK status indicated

  - TRIM value adjusted by one trimming step in AUTOTRIM mode
- When the frequency error is above or equal to the warning limit but below the error limit, a stronger trimming action is necessary, and there is a risk that the optimal TRIM value is not reached for the next period.
  - SYNCWARN status indicated
  - TRIM value adjusted by two trimming steps in AUTOTRIM mode
- When the frequency error is above or equal to the error limit, the frequency is out of the trimming range. This can also happen when the SYNC input is not clean, or when some SYNC pulse is missing (for example when one USB SOF is corrupted).
  - SYNCERR or SYNCMISS status indicated
  - TRIM value not changed in AUTOTRIM mode

Note: If the actual value of the TRIM field is close to its limits and the automatic trimming can force it to overflow or underflow, the TRIM value is set to the limit, and the TRIMOVF status is indicated.

In AUTOTRIM mode (AUTOTRIMEN bit set in the CRS_CR register), the TRIM field of CRS_CR is adjusted by hardware and is read-only.

8.4.6 CRS initialization and configuration

RELOAD value

The RELOAD value must be selected according to the ratio between the target frequency and the frequency of the synchronization source after prescaling. This value is decreased by 1, to reach the expected synchronization on the zero value. The formula is the following:

$$RELOAD = (f_{TARGET} / f_{SYNC}) - 1$$

The reset value of the RELOAD field corresponds to a target frequency of 48 MHz and a synchronization signal frequency of 1 kHz (SOF signal from USB).

FELIM value

The selection of the FELIM value is closely coupled with the HSI48 oscillator characteristics and its typical trimming step size. The optimal value corresponds to half of the trimming step size, expressed as a number of oscillator clock ticks. The following formula can be used:

$$FELIM = (f_{TARGET} / f_{SYNC}) * STEP[\%] / 100\% / 2$$

The result must be always rounded up to the nearest integer value to obtain the best trimming response. If frequent trimming actions are not needed in the application, the hysteresis can be increased by slightly increasing the FELIM value.

The reset value of the FELIM field corresponds to $(f_{TARGET} / f_{SYNC}) = 48000$, and to a typical trimming step size of 0.14%.

Note: The trimming step size depends upon the product, check the datasheet for accurate setting.

Caution: There is no hardware protection from a wrong configuration of the RELOAD and FELIM fields, this can lead to an erratic trimming response. The expected operational mode requires proper setup of the RELOAD value (according to the synchronization source frequency), which is also greater than 128 * FELIM value (OUTRANGE limit).

8.5 CRS in low-power modes

Table 71. Effect of low-power modes on CRS

Mode	Description
Sleep	No effect. CRS interrupts cause the device to exit the Sleep mode.
Stop	CRS registers are frozen. The CRS stops operating until the Stop mode is exited and the HSI48 oscillator is restarted.
Standby	The peripheral is powered down and must be reinitialized after exiting Standby mode.
Shutdown	The peripheral is powered down and must be reinitialized after exiting Shutdown mode.

8.6 CRS interrupts

Table 72. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Clear flag bit
Expected synchronization	ESYNCF	ESYNCIE	ESYNCC
Synchronization OK	SYNCOKF	SYNCOKIE	SYNCOKC
Synchronization warning	SYNCWARNF	SYNCWARNIE	SYNCWARNC
Synchronization or trimming error (TRIMOVF, SYNCMISS, SYNCERR)	ERRF	ERRIE	ERRC

8.7 CRS registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed only by words (32-bit).

8.7.1 CRS control register (CRS_CR)

Address offset: 0x00

Reset value: 0x0000 2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	TRIM[5:0]						SW SYNC	AUTO TRIMEN	CEN	Res.	ESYNC IE	ERR IE	SYNC WARNIE	SYNC OKIE
		rw	rw	rw	rw	rw	rw	rt_w1	rw	rw		rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **TRIM[5:0]**: HSI48 oscillator smooth trimming

These bits provide a user-programmable trimming value to the HSI48 oscillator. They can be programmed to adjust to variations in voltage and temperature that influence the oscillator frequency.

The default value of the HSI48 oscillator smooth trimming is 32, for a complete TRIM range of 64 steps.

The trimming step is specified in the product datasheet. A higher TRIM value corresponds to a higher output frequency.

When the AUTOTRIMEN bit is set, this field is controlled by hardware and is read-only.

Bit 7 **SWSYNC**: Generate software SYNC event

This bit is set by software in order to generate a software SYNC event. It is automatically cleared by hardware.

0: No action

1: A software SYNC event is generated.

Bit 6 **AUTOTRIMEN**: Automatic trimming enable

This bit enables the automatic hardware adjustment of TRIM bits according to the measured frequency error between two SYNC events. If this bit is set, the TRIM bits are read-only. The TRIM value can be adjusted by hardware by one or two steps at a time, depending on the measured frequency error value. Refer to *Section 8.4.5* for more details.

0: Automatic trimming disabled, TRIM bits can be adjusted by the user.

1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.

Bit 5 **CEN**: Frequency error counter enable

This bit enables the oscillator clock for the frequency error counter.

0: Frequency error counter disabled

1: Frequency error counter enabled

When this bit is set, the CRS_CFGR register is write-protected and cannot be modified.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **ESYNCIE**: Expected SYNC interrupt enable

0: Expected SYNC (ESYNCF) interrupt disabled

1: Expected SYNC (ESYNCF) interrupt enabled

Bit 2 **ERRIE**: Synchronization or trimming error interrupt enable

0: Synchronization or trimming error (ERRF) interrupt disabled

1: Synchronization or trimming error (ERRF) interrupt enabled

Bit 1 **SYNCWARNIE**: SYNC warning interrupt enable

0: SYNC warning (SYNCWARNF) interrupt disabled

1: SYNC warning (SYNCWARNF) interrupt enabled

Bit 0 **SYNCOKIE**: SYNC event OK interrupt enable

0: SYNC event OK (SYNCOKF) interrupt disabled

1: SYNC event OK (SYNCOKF) interrupt enabled

8.7.2 CRS configuration register (CRS_CFGR)

This register can be written only when the frequency error counter is disabled (the CEN bit is cleared in CRS_CR). When the counter is enabled, this register is write-protected.

Address offset: 0x04

Reset value: 0x2022 BB7F

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

Bit 31 **SYNCPOL**: SYNC polarity selection

This bit is set and cleared by software to select the input polarity for the SYNC signal source.

0: SYNC active on rising edge (default)

1: SYNC active on falling edge

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **SYNCSRC[1:0]**: SYNC signal source selection

These bits are set and cleared by software to select the SYNC signal source (see *Section 8.4.2: CRS internal signals*).

00: crs_sync_in_0 selected as SYNC signal source

01: crs_sync_in_1 selected as SYNC signal source

10: crs_sync_in_2 selected as SYNC signal source

11: crs_sync_in_3 selected as SYNC signal source

Note: When using USB LPM (link power management) and the device is in Sleep mode, the periodic USB SOF is not generated by the host. No SYNC signal is therefore provided to the CRS to calibrate the 48 MHz clock on the run. To guarantee the required clock precision after waking up from Sleep mode, the LSE clock or the SYNC pin must be used as SYNC signal.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **SYNCDIV[2:0]**: SYNC divider

These bits are set and cleared by software to control the division factor of the SYNC signal.

000: SYNC not divided (default)

001: SYNC divided by 2

010: SYNC divided by 4

011: SYNC divided by 8

100: SYNC divided by 16

101: SYNC divided by 32

110: SYNC divided by 64

111: SYNC divided by 128

Bits 23:16 **FELIM[7:0]**: Frequency error limit

FELIM contains the value to be used to evaluate the captured frequency error value latched in the FECAP[15:0] bits of the CRS_ISR register. Refer to *Section 8.4.5* for more details about FECAP evaluation.

Bits 15:0 **RELOAD[15:0]**: Counter reload value

RELOAD is the value to be loaded in the frequency error counter with each SYNC event. Refer to *Section 8.4.4* for more details about counter behavior.

8.7.3 CRS interrupt and status register (CRS_ISR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FECAP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEDIR	Res.	Res.	Res.	Res.	TRIM OVF	SYNC MISS	SYNC ERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNC WARNF	SYNC OKF
r					r	r	r					r	r	r	r

Bits 31:16 **FECAP[15:0]**: Frequency error capture

FECAP is the frequency error counter value latched in the time of the last SYNC event. Refer to *Section 8.4.5* for more details about FECAP usage.

Bit 15 **FEDIR**: Frequency error direction

FEDIR is the counting direction of the frequency error counter latched in the time of the last SYNC event. It shows whether the actual frequency is below or above the target.

0: Up-counting direction, the actual frequency is above the target

1: Down-counting direction, the actual frequency is below the target

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **TRIMOVF**: Trimming overflow or underflow

This flag is set by hardware when the automatic trimming tries to over- or under-flow the TRIM value. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No trimming error signaled

1: Trimming error signaled

Bit 9 **SYNCMISS**: SYNC missed

This flag is set by hardware when the frequency error counter reaches value FELIM * 128 and no SYNC is detected, meaning either that a SYNC pulse was missed, or the frequency error is too big (internal frequency too high) to be compensated by adjusting the TRIM value, hence some other action must be taken. At this point, the frequency error counter is stopped (waiting for a next SYNC), and an interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No SYNC missed error signaled

1: SYNC missed error signaled

Bit 8 **SYNCERR**: SYNC error

This flag is set by hardware when the SYNC pulse arrives before the ESYNC event and the measured frequency error is greater than or equal to FELIM * 128. This means that the frequency error is too big (internal frequency too low) to be compensated by adjusting the TRIM value, and that some other action has to be taken. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No SYNC error signaled

1: SYNC error signaled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCF**: Expected SYNC flag

This flag is set by hardware when the frequency error counter reached a zero value. An interrupt is generated if the ESYNCIE bit is set in the CRS_CR register. It is cleared by software by setting the ESYNCC bit in the CRS_ICR register.

0: No expected SYNC signaled

1: Expected SYNC signaled

Bit 2 **ERRF**: Error flag

This flag is set by hardware in case of any synchronization or trimming error. It is the logical OR of the TRIMOVF, SYNCMISS and SYNCERR bits. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software in reaction to setting the ERRC bit in the CRS_ICR register, which clears the TRIMOVF, SYNCMISS and SYNCERR bits.

0: No synchronization or trimming error signaled

1: Synchronization or trimming error signaled

Bit 1 **SYNCWARNF**: SYNC warning flag

This flag is set by hardware when the measured frequency error is greater than or equal to FELIM * 3, but smaller than FELIM * 128. This means that to compensate the frequency error, the TRIM value must be adjusted by two steps or more. An interrupt is generated if the SYNCWARNIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCWARNC bit in the CRS_ICR register.

0: No SYNC warning signaled

1: SYNC warning signaled

Bit 0 **SYNCOKF**: SYNC event OK flag

This flag is set by hardware when the measured frequency error is smaller than FELIM * 3. This means that either no adjustment of the TRIM value is needed or that an adjustment by one trimming step is enough to compensate the frequency error. An interrupt is generated if the SYNCOKIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCOKC bit in the CRS_ICR register.

0: No SYNC event OK signaled

1: SYNC event OK signaled

8.7.4 CRS interrupt flag clear register (CRS_ICR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

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

Bit 3 **ESYNCC**: Expected SYNC clear flag
Writing 1 to this bit clears the ESYNCF flag in the CRS_ISR register.

Bit 2 **ERRC**: Error clear flag
Writing 1 to this bit clears TRIMOVF, SYNCMISS, and SYNCERR bits and consequently also the ERRF flag in the CRS_ISR register.

Bit 1 **SYNCWARNC**: SYNC warning clear flag
Writing 1 to this bit clears the SYNCWARNF flag in the CRS_ISR register.

Bit 0 **SYNCOKC**: SYNC event OK clear flag
Writing 1 to this bit clears the SYNCOKF flag in the CRS_ISR register.

8.7.5 CRS register map

Table 73. CRS register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRS_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]						SWSYNC	AUTOTRIMEN	CEN	Res.	ESYNCIE	ERRIE	SYNCWARNIE	SYNCOKIE
	Reset value	0x0000 2000																															
0x04	CRS_CFGR	SYNCPOL	Res.	SYNCSRC[1:0]		Res.	SYNCDIV[2:0]			FELIM[7:0]								RELOAD[15:0]															
	Reset value	0		1	0		0	0	0	0	0	1	0	0	0	1	0	1	0	1	1	1	0	1	1	0	1	1	1	1	1	1	1
0x08	CRS_ISR	FECAP[15:0]																FEDIR	Res.	Res.	Res.	Res.	TRIMOVF	SYNCMISS	SYNCERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNCWARNF	SYNCOKF
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0					0	0	0	0
0x0C	CRS_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESYNCC	ERRC	SYNCWARNC	SYNCOKC
	Reset value																													0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

9 General-purpose I/Os (GPIO)

9.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 16-bit reset register (GPIOx_BRR) and a 32-bit set/reset register (GPIOx_BSRR).

In addition, all GPIOs have a 32-bit locking register (GPIOx_LCKR), two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

9.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_ BSRR) for bitwise write access to GPIOx_ODR
- Lock mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

9.3 GPIO functional description

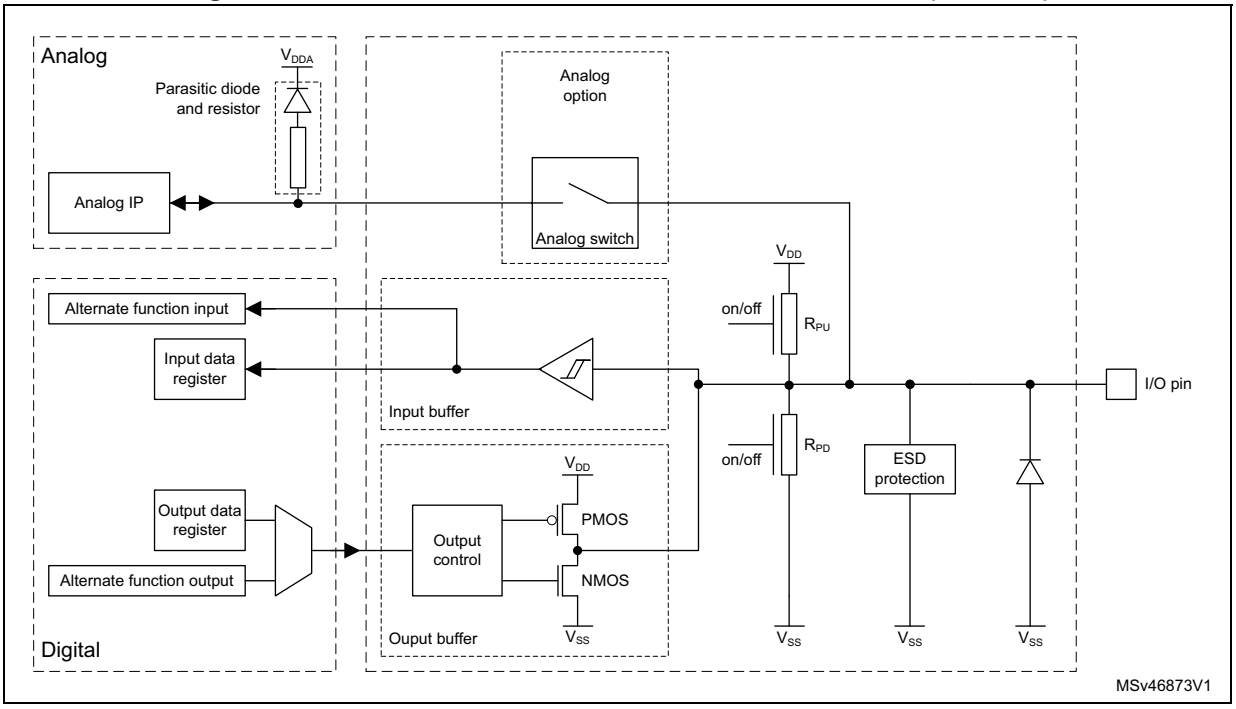
Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers must be accessed as 32-bit words, half-words, or bytes. The GPIOx_BSRR and GPIOx_BRR registers allow atomic read/modify accesses to any of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

The following figure shows the basic structure of a three-volt or five-volt tolerant GPIO (TT or FT). The *Table 74* gives the possible port bit configurations.

Figure 70. Structure of three-volt or five-volt tolerant GPIO (TT or FT)



Note: On a TT GPIO, the analog switch is not present and replaced by a direct connection. The analog bloc parasitic circuitry does not allow five-volt tolerance.

Table 74. Port bit configuration(1)

MODE(i) [1:0]	OTYPE(i)	OSPEED(i) [1:0]	PUPD(i) [1:0]		I/O configuration	
01	0	SPEED [1:0]	0	0	GP output	PP
	0		0	1	GP output	PP + PU
	0		1	0	GP output	PP + PD
	0		1	1	Reserved	
	1		0	0	GP output	OD
	1		0	1	GP output	OD + PU
	1		1	0	GP output	OD + PD
	1		1	1	Reserved (GP output OD)	

Table 74. Port bit configuration(1) (continued)

MODE(i) [1:0]	OTYPE(i)	OSPEED(i) [1:0]		PUPD(i) [1:0]		I/O configuration	
10	0	SPEED [1:0]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0	Input/output	Analog, PD
	x	x	x	1	1	Reserved	

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

9.3.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in analog mode.

The debug pins are in AF pull-up/pull-down after reset:

- PA15: JTDI in pull-up
- PA14: JTCK/SWCLK in pull-down
- PA13: JTMS/SWDIO in pull-up
- PB4: NJTRST in pull-up
- PB3: JTDO/TRACESWO in floating state no pull-up/pull-down

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, the high level is high-Z).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors that can be activated or not depending on the value in the GPIOx_PUPDR register.

9.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there is no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to 16 alternate function inputs (AF0 to AF15) that can be configured through the GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15) registers:

- After reset, the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages. The I/O pins on port P and port O do not have alternate functions as these are dedicated for the XSPI interface.

To use an I/O in a given configuration, the user must proceed as follows:

- **Debug function**: after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host.
- **GPIO**: configure the desired I/O as output, input or analog in the GPIOx_MODER register.
- **Peripheral alternate function**:
  - Connect the I/O to the desired AFx in one of the GPIOx_AFRL or GPIOx_AFRH register.
  - Select the type, pull-up/pull-down, and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDR registers respectively.
  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
- **Additional functions:**
  - For all ADCs, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC registers.
  - For the additional functions like RTC, WKUPx, and oscillators, configure the required function in the related RTC, PWR, and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

Refer to the “Alternate function mapping” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

9.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os. The GPIOx_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

9.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (*GPIO port input data register (GPIOx_IDR) (x = A to I)* and *GPIO port output data register (GPIOx_ODR) (x = A to I)*).

GPIOx_ODR stores the data to be output, it is read/write accessible. The data input through the I/O is stored into the input data register (GPIOx_IDR), a read-only register.

9.3.5 I/O data bitwise handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register that allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). The bit set reset register has twice the size of GPIOx_ODR.

To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BS(i) and BR(i). When written to 1, BS(i) sets the corresponding ODR(i) bit. When written to 1, BR(i) resets the ODR(i) corresponding bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using the GPIOx_BSRR register to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can always be accessed directly. The GPIOx_BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: one or more bits can be modified in a single atomic AHB write access.

9.3.6 GPIO locking mechanism

The GPIO control registers can be frozen by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL, GPIOx_AFRH.

To write the GPIOx_LCKR register, a specific write/read sequence must be applied. When the right LOCK sequence is applied to the bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence is applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

The LOCK sequence can only be performed using a word (32-bit long) access to the GPIOx_LCKR register because GPIOx_LCKR bit 16 must be set at the same time as the [15:0] bits.

9.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, the user can connect an alternate function to some other pin as required by the application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using the GPIOx_AFRL and GPIOx_AFRH alternate function registers. The application can

thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the device datasheet.

9.3.8 External interrupt/wake-up lines

All ports have external interrupt capability. To use external interrupt lines, the port can be configured in input, output, or alternate function mode (the port must not be configured in analog mode). Refer to *Section 16: Extended interrupt and event controller (EXTI)*.

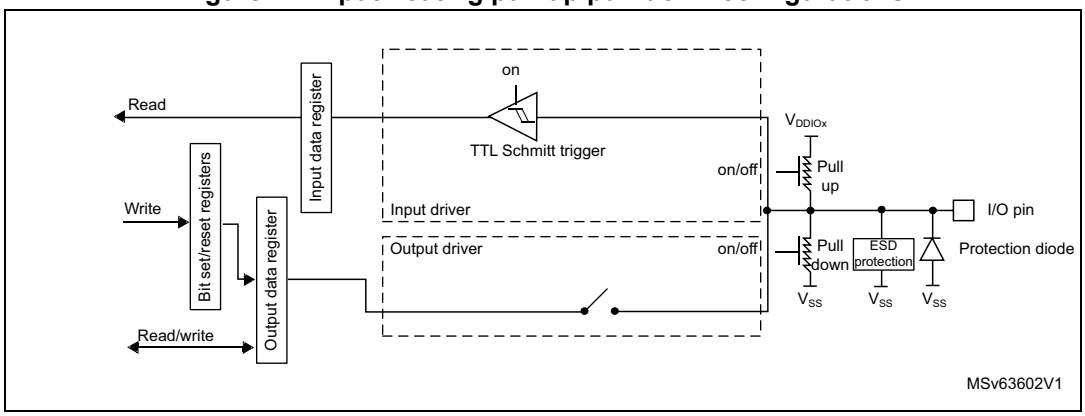
9.3.9 Input configuration

When the I/O port is programmed as input:

- The output buffer is disabled.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register provides the I/O state.

The figure below shows the input configuration of the I/O port bit.

Figure 71. Input floating/pull-up/pull-down configurations



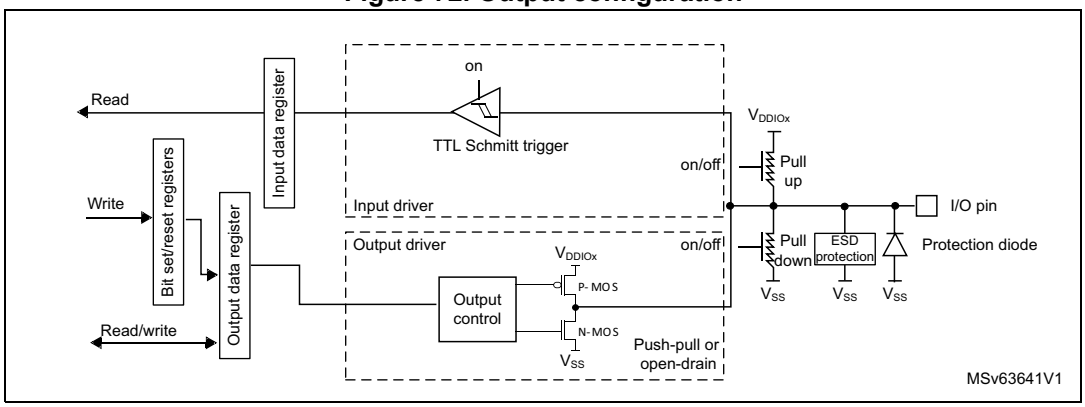
9.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open-drain mode: a 0 in the output register activates the N-MOS whereas a 1 in the output register leaves the port in high-Z (the P-MOS is never activated).
  - Push-pull mode: a 0 in the output register activates the N-MOS whereas a 1 in the output register activates the P-MOS.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.
- A read access to the output data register gets the last written value.

The figure below shows the output configuration of the I/O port bit.

Figure 72. Output configuration



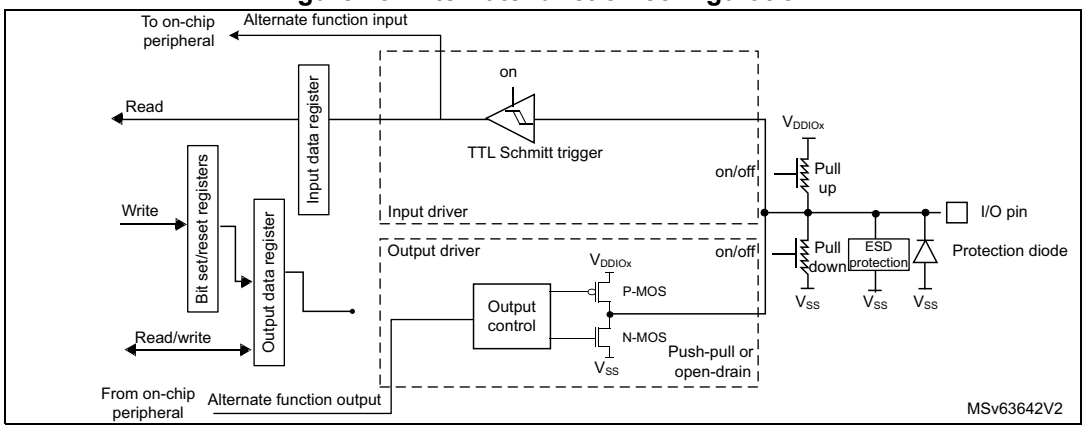
9.3.11 Alternate function configuration

When the I/O port is programmed as an alternate function:

- The output buffer can be configured in open-drain or push-pull mode.
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data).
- The Schmitt trigger input is activated.
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.

The figure below shows the alternate function configuration of the I/O port bit.

Figure 73. Alternate function configuration



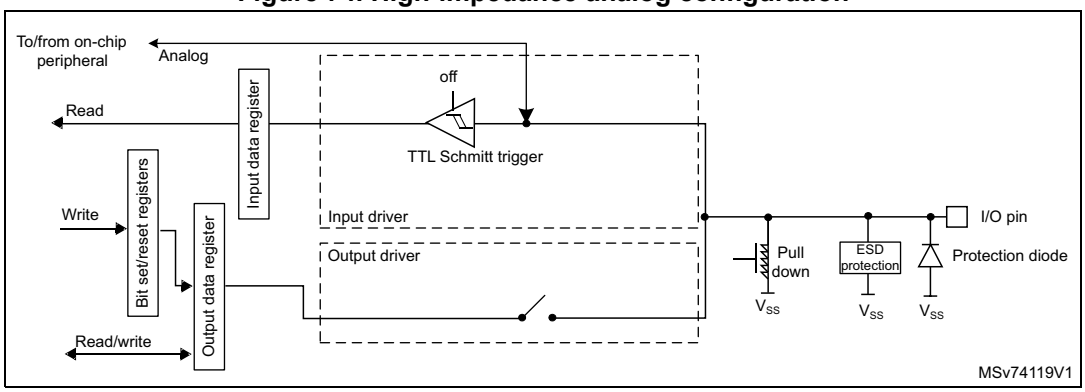
9.3.12 Analog configuration

When the I/O port is programmed as an analog configuration:

- The output buffer is disabled.
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up is disabled by hardware. The weak pull-down is configurable.
- Read access to the input data register gets the value 0.

The figure below shows the high-Z, analog-input configuration of the I/O port bits.

Figure 74. High-impedance analog configuration



9.3.13 Using the HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched off (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched on (by setting the HSEON or LSEON bit in the RCC_CSR register), the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect.

When the oscillator is configured in a user external clock mode, only the pin is reserved for clock input, and the OSC_OUT or OSC32_OUT pin can still be used as normal GPIO.

9.3.14 Using the GPIO pins in the RTC supply domain

The PC13/PC14/PC15 GPIO functionality is lost when the core supply domain is powered off (when the device enters Standby mode). In this case, if their GPIO configuration is not bypassed by the RTC configuration, these pins are set in an analog input mode.

For details about I/O control by the RTC, refer to *Section 42.3: RTC functional description*

9.3.15 Privileged and unprivileged modes

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

9.3.16 High-speed low-voltage mode (HSLV)

The I/Os can increase their maximum speed at low voltage by configuring them in HSLV mode. The I/O HSLV bit controls whether the I/O output speed is optimized to operate at 3.3 V (default setting) or at 1.8 V (HSLV = 1). The CSI must be enabled to configure the I/Os in HSLV mode.

Procedure to set the HSLV mode:

1. Set the user option byte for the GPIO (VDDIO_HSLV) in the *FLASH option byte word 1 status register (FLASH_OBW1SR)*
2. Activate the HSLV (IOHSLV) in the *SBS I/O compensation cell control and status register (SBS_CCCSR)*.

Caution: The I/O HSLV configuration bit must not be set if the I/O supply (V_{DD}) is above 2.7 V. Setting it while the voltage is higher than 2.7 V can damage the device.

There is no hardware protection associated with this feature so it is recommended to use it only as a static configuration for fixed I/O supply.

Caution: The GPIOs are all programed with the same HSLV setting.

GPIOs that are shared with the USB are limited in terms of frequency, to hundreds of kHz for UCPD-based GPIOs, and up to 1 MHz for USBFS-based GPIOs. Refer to the datasheet for the exact values. In order to be used, the I/O supply must be above 2.7 V for the USBFS.

9.3.17 I/O compensation cell

The I/O commutation slew rate (t_{fall} / t_{rise}) can be adapted by software depending on process, voltage, and temperature conditions, to reduce the I/O noise on the power supply. The I/O compensation can be enabled and applied in the *SBS I/O compensation cell control and status register (SBS_CCCSR)*.

9.4 GPIO registers

This section gives a detailed description of the GPIO registers.

The peripheral registers can be written in word, half-word or byte mode.

9.4.1 GPIO port mode register (GPIOx_MODER) (x = A to I)

Address offset: 0x00

Reset value: 0xABFF FFFF (port A)

Reset value: 0xFFFF FEBF (port B)

Reset value: 0xFFFF FFFF (ports C...H)

Reset value: 0x0003 03FF (port I)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MODEy[1:0]:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O mode.

00: Input mode

01: General purpose output mode

10: Alternate function mode

11: Analog mode (reset state)

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to I)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTy:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output type.

0: Output push-pull (reset state)

1: Output open-drain

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to I)

Address offset: 0x08

Reset value: 0x0C00 0000 (for port A)

Reset value: 0x0000 00C0 (for port B)

Reset value: 0x0000 0000 (for the other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSPEEDy[1:0]**: Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output speed.

00: Low speed

01: Medium speed

10: High speed

11: Very-high speed

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to I)

Address offset: 0x0C

Reset value: 0x6400 0000 (for port A)

Reset value: 0x0000 0100 (for port B)

Reset value: 0x0000 0000 (for the other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PUPDy[1:0]:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O pull-up or pull-down

00: No pull-up, pull-down

01: Pull-up

10: Pull-down

11: Reserved

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.4.5 GPIO port input data register (GPIOx_IDR) (x = A to I)

Address offset: 0x10

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDy:** Port x input data I/O pin y (y = 15 to 0)

These bits are read-only. They contain the input value of the corresponding I/O port.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.4.6 GPIO port output data register (GPIOx_ODR) (x = A to I)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODy:** Port output data I/O pin y (y = 15 to 0)

These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A to I).

The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to I)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODy bit

1: Resets the corresponding ODy bit

Note: If both BSy and BRy are set, BSy has priority.

The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Bits 15:0 **BSy:** Port x set I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODy bit

1: Sets the corresponding ODy bit

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to I)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

- LOCK key write sequence:
  WR LCKR[16] = 1 + LCKR[15:0]
  WR LCKR[16] = 0 + LCKR[15:0]
  WR LCKR[16] = 1 + LCKR[15:0]
- LOCK key read:
  RD LCKR[16] = 1 (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.
Any error in the lock sequence aborts the LOCK.
After the first LOCK sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:0 **LCKy:** Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is 0

0: Port configuration not locked

1: Port configuration locked

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to I)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELy[3:0]:** Alternate function selection for port x I/O pin y (y = 7 to 0)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to I)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELy[3:0]:** Alternate function selection for port x I/O pin y (y = 15 to 8)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to I)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRy:** Port x reset IO pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODy bit

1: Reset the corresponding ODy bit

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.4.12 GPIO register map

Table 75. GPIO register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOx_MODER (x = A to I)	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value for port A	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value for port B	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1
	Reset value for ports C...I	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	GPIOx_OTYPER (x = A to I)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOx_ OSPEEDR (x = A to I)	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value for port A	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value for port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
	Reset value for ports C...I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOx_PUPDR (x = A to I)	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value for port A	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value for port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	Reset value for ports C...I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	GPIOx_IDR (x = A to I)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x14	GPIOx_ODR (x = A to I)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	GPIOx_BSRR (x = A to I)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	GPIOx_LCKR (x = A to I)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	GPIOx_AFRL (x = A to I)	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 75. GPIO register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	GPIOx_AFRH (x = A to I)	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]				AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	GPIOx_BRR (x = A to I)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

10 System configuration, boot and security (SBS)

10.1 SBS introduction

The STM32H7Pxxx devices feature a set of configuration registers located in the SBS. On top of various device configurations, this SBS peripheral controls key boot and security features, including debug.

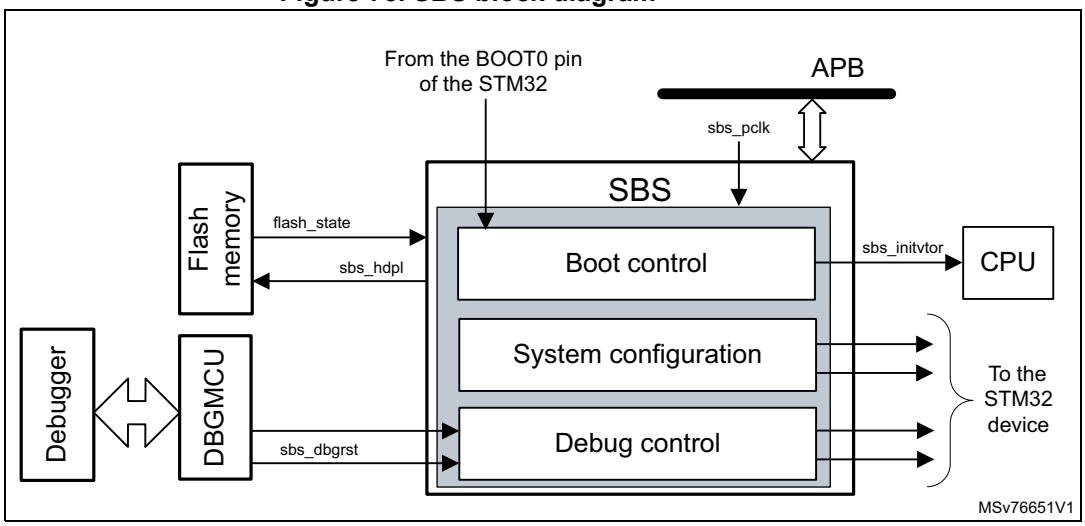
10.2 SBS main features

- Boot control
  - Upon system reset, configure the Cortex-M7 boot address and the hide protection level (HDPL) depending on the device life cycle (open or close).
- Debug control
  - Control the opening of the device debug interface, ensuring the sequencing of events that guarantee the device security
- System configuration
  - Analog switch configuration management
  - I/O compensation cell management
  - Selection of source input to the EXTI inputs used for external interrupt/event detection
  - Configuration for Cortex-M7 FPU interrupts
- Security status
  - BKPRAM and AXISRAM1 erase
- Register for applications to pass on a command to the RSS (root secure services), executed at the next reset

10.3 SBS functional description

10.3.1 SBS block diagram

Figure 75. SBS block diagram



10.3.2 SBS internal signals

The tables below detail the SBS pins and internal signals.

Table 76. SBS input/output pins

Pin name	Type	Description
BOOT0	Input	Boot from flash memory or bootloader.

Table 77. SBS internal input/output signals

Signal name	Type	Description
sbs_pclk	Input	APB bus clock
sbs_dbgrst		Debug configuration reset from DBGMCU
flash_state		Nonvolatile security status input from flash memory
sbs_hdpl	Output	Hide protection level information
sbs_initvtor		Vector address for Cortex-M7. See *Section 10.3.5* for more details.

10.3.3 SBS reset and clocks

The SBS configuration port is clocked by the APB bus clock. There is a general reset and a debug configuration reset controlled in DBGMCU.

10.3.4 SBS hide protection management

The HDPL is a monotonic counter incremented during the boot stages. It is reset to its default value only following a power-on reset or a system reset. This default value (0 or 1) depends on the device lifecycle, as defined in *Table 78: SBS boot control*.

The device uses HDPL information to automatically isolate code and its associated secrets (like keys) during the boot process. Incrementing HDPL ensures that private code and data for one boot stage cannot be directly accessible from later boot stages.

Note: The use of HDPL in the embedded flash memory is detailed in Section 5: Embedded flash memory (FLASH).

The HDPL value can be set between 0 and 3. When reaching the value 3, the HDPL keeps this value until reset. The current HDPL value is readable in the HDPL bitfield of SBS_HDPLSR register.

Incrementing HDPL

The application must write 0x6A to the INCR_HDPL bitfield in the SBS_HDPLCR register to increment the HDPL by one.

After such increment, and before doing any subsequent action, the application must check if the HDPL has effectively been incremented, by reading SBS_HDPLSR.

10.3.5 SBS boot control

Figure 76 and *Table 78* show how the SBS uses multiple information to define the Cortex-M7 boot address (INITVTOR) and the HDPL before the CPU reset is released.

Figure 76. SBS boot control

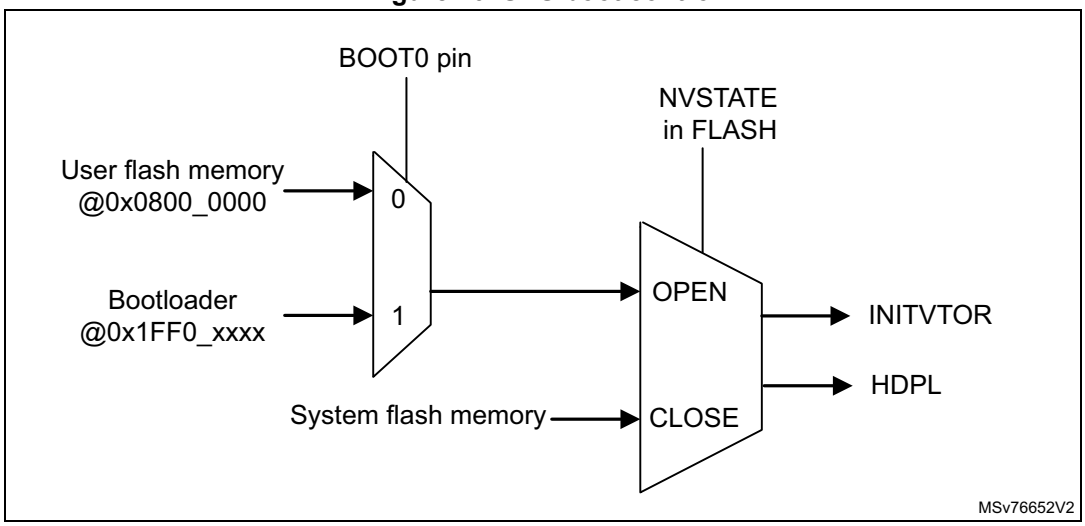


Table 78. SBS boot control

SBS input		SBS output		
NVSTATE (from FLASH)	**BOOT0 pin**	**Boot address**	**INITVTOR(1) (to Cortex-M7)**	**HDPL value in SBS_HDPLSR**
NVOPEN	0	User flash memory	0x0800 0000	1
	1	Bootloader(2)	Defined by ST	1
NVCLOSE	x	System flash memory (ST RSS)		0

1. Readable in SBS_BOOTSR register.
2. Located in the system memory. It is used to reprogram the flash memory by using USART, I2C/I3C, SPI, FDCAN, or USB_OTG in device mode through the DFU (device firmware upgrade).

The BOOT0 value from the external pin is latched when the power-on reset is released or when the device exits standby mode. This pin is in input mode during the complete reset phase, and then switches automatically in analog mode after reset is released.

The NVSTATE value comes from the flash memory interface, following an automatic option-byte load sequence (see *Section 5: Embedded flash memory (FLASH)* for details).

10.3.6 SBS debug management

Device debugging may or may not be allowed, depending on the device life cycle:

- For open devices (NVSTATE = NVOPEN), the debug is always allowed.
- For close devices (NVSTATE = NVCLOSE), the debug is disabled at boot with a policy defined by the option byte DBG_AUTH[7:0] in the FLASH_ROTSR. Possible options are:
  - 0xB4: locked device (no debug, no regression)
  - 0x8A: closed device (no debug).
  - 0x6F: closed device (no debug out of reset), with a possibility for user code executing in HDPL1 to open the debug. This user code can deactivate this debug opening feature by writing DBGCFG_LOCK[7:0] to 0xC3.

For both closed device options a regression option to NVOPEN state is possible using the 128-bit regression password provisioned during the manufacturing phase of the final product. Refer to **Life cycle management** in *Section 5: Embedded flash memory (FLASH)* for details on this opening sequence.

When NVSTATE = 0x51 in FLASH_NVSR the mailbox access port, CM7 access port, and CPU debug interface are locked. In this situation, the debugger cannot access the CPU and no effective debug can be done. Refer to *Section 52: Debug infrastructure* for more details.

Debug opening by OEM immutable RoT code

When DBG_AUTH[7:0] = 0x6F in FLASH_ROTSR and DBGCFG_LOCK[7:0] = 0xB4 in SBS_DBGLOCKR the OEM code executing in HDPL1 can program the opening of the Cortex-M7 access port by writing to the AP_UNLOCK[7:0] bitfield in the SBS_DBGCR register. In the same register it can also program the opening of the CPU debug interface by writing to the DBG_UNLOCK[7:0] bitfield.

Debug re-openings are effective only when the HDPL in the SBS_HDPLSR register has a value equal or superior to the value programmed in DBG_AUTH_HDPL of SBS_DBGCR register. If DBG_AUTH_HDPL bitfield is kept at 0x0, the debug never opens (forbidden value).

At any time, the application via DBGMCU can reset the debug configuration in the SBS_DBGCR register (see *Section 10.5.4: SBS debug control register (SBS_DBGCR)* for details).

10.3.7 SBS voltage booster for I/O analog switches

By setting BOOSTEN in the SBS_PMCR register, the application reduces the total harmonic distortion (THD) of the I/O analog switches when the supply voltage is below 2.7 V, to guarantee the same performance as with the full voltage range.

To avoid current consumption due to the booster activation when V_{DDA} < 2.7 V and V_{DD} > 2.7 V, V_{DD} can be selected as supply voltage for the analog switches, by setting BOOSTVDDSEL in the SBS_PMCR register.

In this case, BOOSTEN must be cleared to avoid unexpected power consumption.

Table 79. Analog switches recommended configuration

V_{DDA}	V_{DD}	BOOSTEN	BOOSTVDDSEL	Analog switch supply voltage
≤ 2.7 V	≤ 2.7 V	1	0	Booster
	> 2.7 V	0	1	V_{DD}
> 2.7V		0	0	V_{DDA}

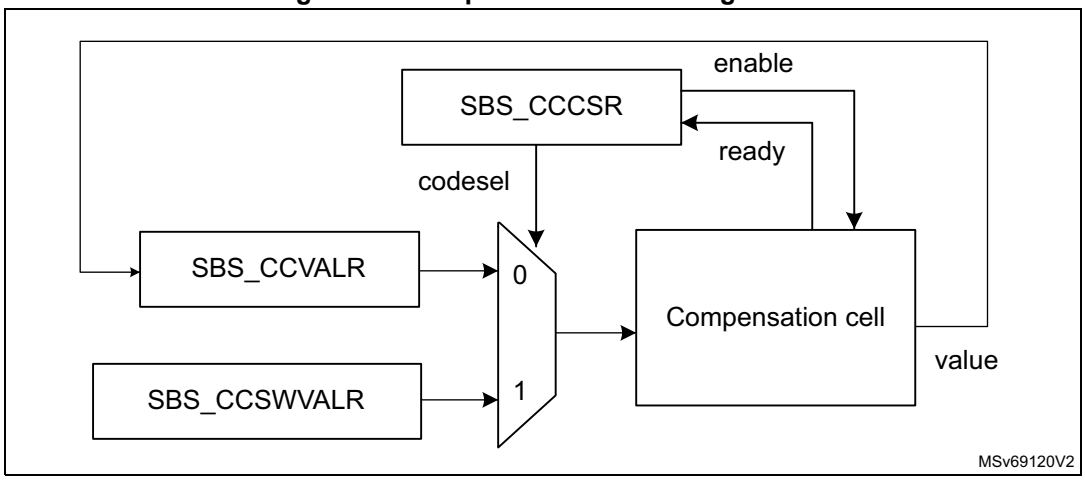
10.3.8 SBS I/O compensation cell management

The I/O compensation cell generates an 8-bit value for the I/O buffer (4 bits for N-MOS and 4 bits for P-MOS), that depends on PVT operating conditions (process, voltage, temperature). These bits are used to control the output impedance in the I/O buffer, and the slew rate of the I/O commutation (the t_{fall} and t_{rise} time), in order to reduce the I/O noise on the power supply.

The CSI is used by the compensation cells and must be enabled before enabling the compensation cells

As shown in *Figure 77*, the compensation cell is split into two blocks: one block to provide an optimal code for the current PVT, and one block to drive the block controlled by the software.

Figure 77. Compensation cell management

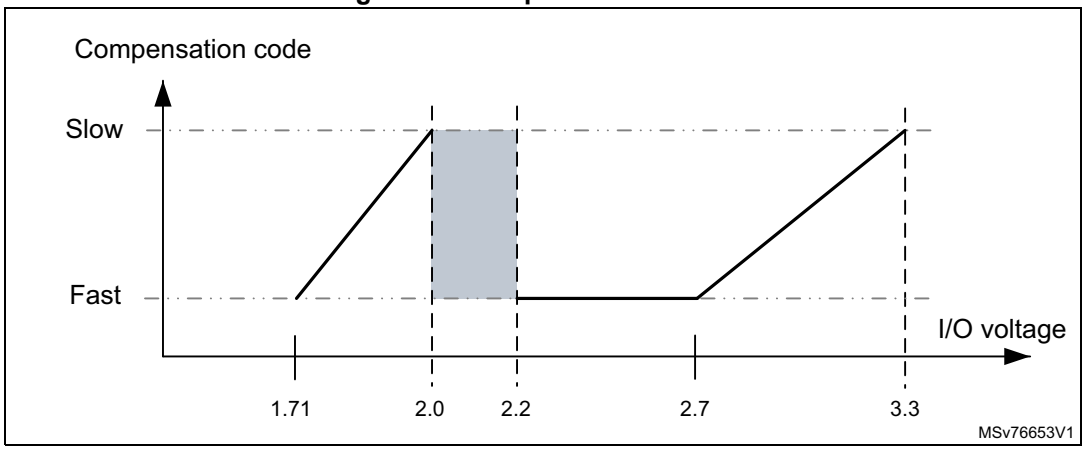


The compensation cell value can be read when the READY flag is set in the SBS_CCCSR register. With CODESEL in SBS_CCCSR register, the application can select the value to apply between the two options: the code from the cell or the code from SBS_CCSWVALR.

One compensation cell is embedded in STM32H7Pxxx devices, for the I/O supplied by V_{DD} or V_{DDA} (see *Section 10.3.7*). It is disabled by default, with a fixed code applied to all the I/Os.

Note: The compensation cell can be used only when 2.7 V ≤ V_{DD}/V_{DDA} ≤ 3.6 V or 1.71 V ≤ V_{DD}/V_{DDA} ≤ 2 V (see following figure).

Figure 78. Compensation cell use



10.3.9 SBS registers access control

Some SBS registers can be accessed only if HDPL = 0 in the SBS_HDPLSR register(hence only accessible by the ST RSS code).

Some bitfields in SBS are security critical, reset in non-secure state (coded as 0xB4) or in secure state (any other values). For bitfields reset in non-secure state, when the secure state is selected writing any value but 0xB4, the bitfield is no more writable until a reset occurs.

As a general rule, it is recommended to write 0xC3 to select a secure state and read back 0x6A to acknowledge the write is effective. The following bitfields implement these protections:

- INCR_HDPL
- DBGCFG_LOCK

10.3.10 SBS error management

The SBS does not manage errors.

10.4 SBS interrupts

The SBS does not support interrupts.

10.5 SBS registers

10.5.1 SBS boot status register (SBS_BOOTSR)

Address offset: 0x000

Reset value: 0xXXXX XXXX

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

Bits 31:0 **INITVTOR[31:0]**: Initial vector for Cortex-M7
This is the physical boot address used by the Cortex-M7 out of reset

10.5.2 SBS hide protection control register (SBS_HDPLCR)

Address offset: 0x010

Reset value: 0x0000 00B4

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

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

Bits 7:0 **INCR_HDPL[7:0]**: Increment HDPL
Write 0x6A to increment device HDPL by one.
After a write, the register value reverts to its default value (0xB4).

10.5.3 SBS hide protection status register (SBS_HDPLSR)

Address offset: 0x014

Reset value: 0x0000 00XX (see *Section 10.3.5*)

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

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

Bits 7:0 **HDPL[7:0]**: Hide protection level
This bitfield returns the current HDPL of the device.
0xB4: HDPL0, corresponding to ST-RSS (default when device is CLOSE)
0x51: HDPL1
0x8A: HDPL2
Others: HDPL3, corresponding to non-boot application

Note: The device nonvolatile state is defined in FLASH_NVSR register of the embedded flash memory.

10.5.4 SBS debug control register (SBS_DBGCR)

Address offset: 0x020

Reset value: 0x0000 0000

Read as zero and write is ignored if HDPL is different from 0 or 1 in SBS_HDPLSR register.

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **DBG_AUTH_HDPL[7:0]**: Authenticated debug hide protection level
Writing to this bitfield defines at which HDPL the debug opens.
0x51: HDPL1
0x8A: HDPL2
0x6F: HDPL3
Note: Writing any other values is ignored. Reading any other value means the debug opening always fails.

Bits 15:8 **DBG_UNLOCK[7:0]**: Debug unlock
Write 0xB4 to this bitfield to open the debug when HDPL in SBS_HDPLSR equals to DBG_AUTH_HDPL in this register.

Bits 7:0 **AP_UNLOCK[7:0]**: Access port unlock
Write 0xB4 to this bitfield to open the device access port.

10.5.5 SBS debug lock register (SBS_DBGLOCKR)

Address offset: 0x024

Reset value: 0x0000 00B4

This register is reset when sbs_dbgrst is asserted.

Read as zero and write is ignored if HDPL is different from 0 or 1 in SBS_HDPLSR register.

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

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

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

Bits 7:0 **DBGCFG_LOCK[7:0]**: Debug configuration lock
Reading this bitfield returns 0x6A if the bitfield value is different from 0xB4.
0xB4: Writes to SBS_DBGCR allowed (default)
Others: Writes to SBS_DBGCR ignored
Note: 0xC3 is the recommended value to lock the debug configuration using this bitfield.

10.5.6 SBS RSS command register (SBS_RSSCMDR)

Address offset: 0x034

Reset value: 0x0000 0000

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RSSCMD[15:0]**: RSS command

The application can use this bitfield to pass on a command to the RSS, executed at the next reset.

10.5.7 SBS product mode and configuration register (SBS_PMCR)

Address offset: 0x100

Reset value: 0x0000 0000

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

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

Bit 9 **BOOSTVDDSEL**: Booster V_{DD} selection

This bit selects the analog switch supply voltage, between V_{DD}, V_{DDA} and booster.
To avoid current consumption due to booster activation when V_{DDA} < 2.7 V and V_{DD} > 2.7 V, V_{DD} can be selected as supply voltage for analog switches. In this case, the BOOSTEN bit must be cleared to avoid unwanted power consumption.
When both $V_{DD\ and}$ V_{DDA} are below 2.7 V, the booster is still needed to obtain full AC performances from the I/O analog switches.
0: V_{DDA} selected as analog switch booster voltage supply (when BOOSTEN bit is cleared)
1: V_{DD} selected as analog switch booster voltage supply

Bit 8 **BOOSTEN**: Booster enable

Set this bit to reduce the THD of the analog switches when the supply voltage is below 2.7 V. guaranteeing the same performance as with the full voltage range. To avoid current consumption due to booster activation when V_{DDA} < 2.7 V and V_{DD} > 2.7 V, V_{DD} can be selected as supply voltage for analog switches by setting BOOSTVDDSEL bit in SBS_PMCR. In this case, the BOOSTEN bit must be cleared to avoid unwanted power consumption.
0: Booster disabled
1: Booster enabled

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

10.5.8 SBS FPU interrupt mask register (SBS_FPUIMR)

Address offset: 0x104

Reset value: 0x0000 001F

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

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

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

Bits 5:0 **FPU_IE[5:0]**: FPU interrupt enable

Set and cleared by software to enable the Cortex-M7 FPU interrupts
xxxxx1: Invalid operation interrupt enabled (xxxxx0 to disable)
xxxx1x: Divide-by-zero interrupt enabled (xxxx0x to disable)
xxx1xx: Underflow interrupt enabled (xxx0xx to disable)
xx1xxx: Overflow interrupt enabled (xx0xxx to disable)
x1xxxx: Input denormal interrupt enabled (x0xxxx to disable)
1xxxxx: Inexact interrupt enabled (0xxxxx to disable), disabled by default

10.5.9 SBS memory erase status register (SBS_MESR)

Address offset: 0x108

Reset value: 0x0000 0000

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

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

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

Bit 1 **AXIRMEF**: AXISRAM1 memory erase flag

This bit is set by hardware when AXISRAM1 erase is ongoing after a power-on reset or a tamper event (see *Section 43: Tamper and backup registers (TAMP)* for details). This bit is set when the erase is done.
0: Automatic erase of AXISRAM1 memory ongoing
1: Automatic erase of AXISRAM1 memory done

Bit 0 **BKPMEF**: Backup SRAM memory erase flag

This bit is set by hardware when BKPRAM erase is ongoing after a power-on reset or a tamper event (see *Section 43: Tamper and backup registers (TAMP)* for details). This bit is set when the erase is done.
0: Automatic erase of BKPRAM memory ongoing
1: Automatic erase of BKPRAM memory done

10.5.10 SBS I/O compensation cell control and status register (SBS_CCCSR)

Address offset: 0x110

Reset value: 0x0000 0000

For more details on this register, refer to *Section 10.3.7* and *Section 10.3.8*.

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

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **IOHSLV**: I/O high speed at low voltage

When this bit is set, the speed of the I/Os is optimized when the device voltage is low.
This bit is active only if VDDIO_HSLV user option bit is set in FLASH. It must be used only if the device supply voltage is below 2.7 V. Setting this bit when V_{DD} is higher than 2.7 V may be destructive.
0: No I/O speed optimization when device voltage is low
1: I/O speed optimized when device voltage is low, if VDDIO_HSLV user option is set (no effect otherwise)

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **COMP_RDY**: Compensation cell ready

This bit provides the status of the compensation cell.
0: I/O compensation cell not ready
1: I/O compensation cell ready. The code value provided by the cell can be used.

Note: The CSI clock is required for the compensation cell to work properly. The compensation cell ready bit (COMP_RDY) is not set if the CSI clock is not enabled.

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

Bit 1 **COMP_CODESEL**: Compensation cell code selection

This bit selects the code to be applied for the I/O compensation cell.
0: Code from the cell (available in the SBS_CCVALR)
1: Code from the SBS software value register (SBS_CCSWVALR)

Bit 0 **COMP_EN**: Compensation cell enable

Set this bit to enable the compensation cell.
0: Compensation cell disabled
1: Compensation cell enabled

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

Address offset: 0x114

Reset value: 0x0000 0088

Reset: system reset

For more details on this register, refer to *Section 10.3.8*.

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

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

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

Bits 7:4 **PSRC[3:0]**: PMOS transistors slew-rate compensation

This bitfield returns the PMOS transistors slew-rate compensation value computed by the cell. It is interpreted to compensate the PMOS transistors slew rate in the functional range if COMP_CODESEL = 0 in SBS_CCCSR register.

Bits 3:0 **NSRC[3:0]**: NMOS transistors slew-rate compensation

This bitfield returns the NMOS transistors slew-rate compensation value computed by the cell. It is interpreted to compensate the NMOS transistors slew rate in the functional range if COMP_CODESEL = 0 in SBS_CCCSR register.

10.5.12 SBS compensation cell for I/Os software value register (SBS_CCSWVALR)

Address offset: 0x118

Reset value: 0x0000 0088

For more details on this register, refer to *Section 10.3.8*.

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

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

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

Bits 7:4 **SW_PSRC[3:0]**: Software PMOS transistors slew-rate compensation

This bitfield returns the PMOS transistors slew-rate compensation value computed by the cell. It is interpreted to compensate the PMOS transistors slew rate in the functional range if COMP_CODESEL = 1 in SBS_CCCSR register.

Bits 3:0 **SW_NSRC[3:0]**: Software NMOS transistors slew-rate compensation

This bitfield returns the NMOS transistors slew-rate compensation value computed by the cell. It is interpreted to compensate the NMOS transistors slew rate in the functional range if COMP_CODESEL = 1 in SBS_CCCSR register.

10.5.13 SBS break lock register (SBS_BKLOCKR)

Address offset: 0x120

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARAM1ECC_BL	ARAM2ECC_BL	ARAM3ECC_BL	ARAM4ECC_BL	Res.	Res.	Res.	Res.
								rs	rs	rs	rs				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ITCMECC_BL	DTCMECC_BL	Res.	Res.	Res.	Res.	DCACHEECC_BL	BKRAMECC_BL	CM7LCKUP_BL	Res.	Res.	FLASHECC_BL	PVD_BL	Res.	Res.
	rs	rs					rs	rs	rs			rs	rs		

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ARAM1ECC_BL**: AXISRAM1 ECC error break lock

Set this bit to enable and lock the connection between AXISRAM1 ECC double error detection flag and break inputs of TIM1/8/15/16/17/20 and HRTIMER1/2 peripherals.
Once set, this bit is cleared only by a system reset.
0: AXISRAM1 ECC double error detection flag disconnected from TIM1/8/5/16/17/20 and HRTIMER1/2 break inputs
1: AXISRAM1 ECC double error detection flag connected to TIM1/8/15/16/17/20 and HRTIMER1/2break inputs

Bit 22 **ARAM2ECC_BL**: AXISRAM2 ECC error break lock

Set this bit to enable and lock the connection between AXISRAM2 ECC double error detection flag and break inputs of TIM1/8/15/16/17/20 and HRTIMER1/2 peripherals.
Once set, this bit is cleared only by a system reset.
0: AXISRAM2 ECC double error detection flag disconnected from TIM1/8/5/16/17/20 and HRTIMER1/2 break inputs
1: AXISRAM2 ECC double error detection flag connected to TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs

Bit 21 **ARAM3ECC_BL**: AXISRAM3 ECC error break lock

Set this bit to enable and lock the connection between AXISRAM3 ECC double error detection flag and break inputs of TIM1/8/15/16/17/20 and HRTIMER1/2 peripherals.
Once set this bit is cleared only by a system reset.
0: AXISRAM3 ECC double error detection flag disconnected from TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs
1: AXISRAM3 ECC double error detection flag connected to TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs

Bit 20 **ARAM4ECC_BL**: AXISRAM4 ECC error break lock

Set this bit to enable and lock the connection between AXISRAM4 ECC double error detection flag and break inputs of TIM1/8/15/16/17/20 and HRTIMER1/2 peripherals.
Once set this bit is cleared only by a system reset.
0: AXISRAM4 ECC double error detection flag disconnected from TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs
1: AXISRAM4 ECC double error detection flag connected to TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **ITCMECC_BL**: ITCM ECC error break lock

Set this bit to enable and lock the connection between ITCM ECC double error detection flag and break inputs of TIM1/8/15/16/17/20 and HRTIMER1/2 peripherals.

Once set, this bit is cleared only by a system reset.

0: ITCM ECC double error detection flag disconnected from TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs

1: ITCM ECC double error detection flag connected to TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs

Bit 13 **DTCMECC_BL**: DTCM ECC error break lock

Set this bit to enable and lock the connection between DTCM ECC double error detection flag and break inputs of TIM1/8/15/16/17/20 and HRTIMER1/2 peripherals.

Once set, this bit is cleared only by a system reset.

0: DTCM ECC double error detection flag disconnected from TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs

1: DTCM ECC double error detection flag connected to TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs

Note: The DTCM0 and DTCM1 are Or'ed to give DTCMECC

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

Bit 8 **DCACHEECC_BL**: D-Cache ECC error break lock

Set this bit to enable and lock the connection between D-Cache ECC double error detection flag and break inputs of TIM1/8/15/16/17/20 and HRTIMER1/2 peripherals.

Once set, this bit is cleared only by a system reset.

0: D-Cache ECC double error detection flag disconnected from TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs

1: Cache ECC double error detection flag connected to TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs

Bit 7 **BKRAMECC_BL**: Backup RAM ECC error break lock

Set this bit to enable and lock the connection between backup RAM ECC double error detection flag and break inputs of TIM1/8/15/16/17/20 and HRTIMER1/2 peripherals.

Once set, this bit is cleared only by a system reset.

0: Backup RAM ECC double error detection flag disconnected from TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs.

1: Backup RAM ECC double error detection flag connected to TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs.

Bit 6 **CM7LCKUP_BL**: Cortex-M7 lockup break lock

Set this bit to enable and lock the connection between the Cortex-M7 lockup (HardFault) output and break inputs of TIM1/8/15/16/17/20 and HRTIMER1/2 peripherals.

Once set, this bit is cleared only by a system reset.

0: Cortex-M7 lockup output disconnected from TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs

1: Cortex-M7 lockup output connected to TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **FLASHECC_BL**: Flash ECC error break lock

Set this bit to enable and lock the connection between embedded flash memory ECC double error detection flag and break inputs of TIM1/8/15/16/17/20 and HRTIMER1/2 peripherals.

Once set, this bit is cleared only by a system reset.

0: FLASH ECC double error detection flag disconnected from TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs

1: FLASH ECC double error detection flag connected to TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs

Bit 2 **PVD_BL**: PVD break lock

This bit is set by software and cleared only by a system reset. it can be used to enable and lock the connection to TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs as well as the PVDE and PLS[2:0] bitfields in the PWR_CR1 register.

Once set, this bit is cleared only by a system reset.

0: PVD interrupt disconnected from TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs. PVDE and PLS[2:0] bitfields can be programmed by the application.

1: PVD output connected to TIM1/8/15/16/17/20 and HRTIMER1/2 break inputs. PVDE and PLS[2:0] bits are read only.

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

10.5.14 SBS external interrupt configuration register x (SBS_EXTICRx)

Address offset: 0x130 + 0x4 * (x-1), (x = 1 to 4)

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC_EXTI{(x-1) * 4 + 3}				PC_EXTI{(x-1) * 4 + 2}				PC_EXTI{(x-1) * 4 + 1}				PC_EXTI{(x-1) * 4}			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **PC_EXTI{(x-1) * 4 + 3}**: Port configuration for EXTI{(x-1) * 4 + 3}

This bitfield selects the source input to the EXTI input {(x-1) * 4 + 3} used for external interrupt/event detection.

0x0: PA{(x-1)*4+3} pin
0x1: PB{(x-1)*4+3} pin
0x2: PC{(x-1)*4+3} pin
0x3: PD{(x-1)*4+3} pin
0x4: PE{(x-1)*4+3} pin
0x5: PF{(x-1)*4+3} pin
0x6: PG{(x-1)*4+3} pin
0x7: PH{(x-1)*4+3} pin
0x8: PI{(x-1)*4+3} pin
Others: reserved

Bits 11:8 **PC_EXTI{(x-1) * 4 + 2}**: Port configuration for EXTI{(x-1) * 4 + 2}

This bitfield selects the source input to the EXTI input {(x-1) * 4 + 2} used for external interrupt/event detection.

0x0: PA{(x-1)*4+2} pin
0x1: PB{(x-1)*4+2} pin
0x2: PC{(x-1)*4+2} pin
0x3: PD{(x-1)*4+2} pin
0x4: PE{(x-1)*4+2} pin
0x5: PF{(x-1)*4+2} pin
0x6: PG{(x-1)*4+2} pin
0x7: PH{(x-1)*4+2} pin
0x8: PI{(x-1)*4+2} pin
Others: reserved

Bits 7:4 **PC_EXTI{(x-1) * 4 + 1}**: Port configuration for EXTI{(x-1) * 4 + 1}
This bitfield selects the source input to the EXTI input {(x-1) * 4 + 1} used for external interrupt/event detection.
0x0: PA{(x-1)*4+1} pin
0x1: PB{(x-1)*4+1} pin
0x2: PC{(x-1)*4+1} pin
0x3: PD{(x-1)*4+1} pin
0x4: PE{(x-1)*4+1} pin
0x5: PF{(x-1)*4+1} pin
0x6: PG{(x-1)*4+1} pin
0x7: PH{(x-1)*4+1} pin
0x8: PI{(x-1)*4+1} pin
Others: reserved

Bits 3:0 **PC_EXTI{(x-1) * 4}**: Port configuration for EXTI{(x-1) * 4}
This bitfield selects the source input to the EXTI input {(x-1) * 4} used for external interrupt/event detection.
0x0: PA{(x-1)*4} pin
0x1: PB{(x-1)*4} pin
0x2: PC{(x-1)*4} pin
0x3: PD{(x-1)*4} pin
0x4: PE{(x-1)*4} pin
0x5: PF{(x-1)*4} pin
0x6: PG{(x-1)*4} pin
0x7: PH{(x-1)*4} pin
0x8: PI{(x-1)*4} pin
Others: reserved

10.5.15 SBS register map

Table 80. SBS register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**SBS_BOOTSR**	INITVOR[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x010	**SBS_HDPLCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INCR_HDPL[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0x014	**SBS_HDPLSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPL[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x018	Reserved																																
0x020	**SBS_DBGCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_AUTH_HDPL[7:0]								DBG_UNLOCK[7:0]								AP_UNLOCK[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	**SBS_DBGLOCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGCFG_LOCK[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0x028 to 0x030	Reserved																																
0x034	**SBS_RSSCMDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSSCMD[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038 to 0x0FC	Reserved																																
0x100	**SBS_PMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOSTVDDSEL	BOOSTEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																							0	0								
0x104	**SBS_FPUIMR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPU_IE[5:0]					
	Reset value																											1	1	1	1	1	1
0x108	**SBS_MESR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXISMEF	BKPMEF
	Reset value																															0	0
0x110	**SBS_CCCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOHSLV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP_RDY	Res.	Res.	Res.	Res.	Res.	COMP_CODESE	COMP_EN
	Reset value																0								0						0	0
0x114	**SBS_CCVALR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSRC[3:0]				NSRC[3:0]			
	Reset value																									1	0	0	0	1	0	0	0
0x118	**SBS_ CCSWVALR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW_PSRC [3:0]				SW_NSRC [3:0]			
	Reset value																									1	0	0	0	1	0	0	0
0x120	**SBS_BKLOCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARAM1ECC_BL	ARAM2ECC_BL	ARAM3ECC_BL	ARAM4ECC_BL	Res.	Res.	Res.	Res.	Res.	ITCMECC_BL	DTCMECC_BL	Res.	Res.	Res.	Res.	DCACHEECC_BL	BKRAMECC_BL	CM7LCKUP_BL	Res.	Res.	FLASHECC_BL	PVD_BL	Res.	Res.
	Reset value									0	0	0	0						0	0					0	0	0			0	0		

Table 80. SBS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024 to 0x02C	Reserved																																
0x130 + 0x004 * (x -1) (x=1 to 4)	**SBS_EXTICRx**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC_EXTI {(x-1) * 4 + 3}				PC_EXTI {(x-1) * 4 + 2}				PC_EXTI {(x-1) * 4 + 1}				PC_EXTI {(x-1) * 4}			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

11 Peripheral interconnect

11.1 Introduction

Several peripherals have direct connections between them. This allows autonomous communication and/or synchronization between peripherals, saving CPU resources, thus power supply consumption.

In addition, these hardware connections remove software latency and allow the design of a predictable system.

Depending on peripherals, these interconnections can operate in various power modes: Run, Sleep, or Stop modes.

11.2 Connection summary

Table 81. Peripheral interconnect matrix (1)

Source	Destination																												
	HRTIM1	HRTIM2	TIM1	TIM2	TIM3	TIM4	TIM5	TIM8	TIM9	TIM10	TIM12	TIM13	TIM15	TIM16	TIM17	TIM20	LPTIM1	LPTIM2	LPTIM3	LPTIM4	LPTIM5	ADC1/2	ADC3/4	ADF	GP/HPDMA	TAMP	RTC	HSP	NPU
HRTIM1	-	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	-	*1*	-	*1*	-	-	*1*	-	-	-	-	-	*2*	*2*	-	-	-	-	-	-
HRTIM2	*1*	-	*1*	*1*	*1*	*1*	*1*	*1*	*1*	-	*1*	-	*1*	-	-	*1*	-	-	-	-	-	*2*	*2*	-	-	-	-	-	-
TIM1	*1*	*1*	-	*1*	*1*	*1*	*1*	*1*	*1*	-	*1*	-	*1*	-	-	*1*	-	-	-	-	-	*2*	*2*	-	*12*	-	-	*19*	-
TIM2	*1*	*1*	*1*	-	*1*	*1*	*1*	*1*	*1*	-	*1*	-	*1*	-	-	*1*	-	-	-	-	-	*2*	*2*	-	*12*	-	-	*19*	-
TIM3	*1*	*1*	*1*	*1*	-	*1*	*1*	*1*	*1*	-	*1*	-	*1*	-	-	*1*	-	-	-	-	-	*2*	*2*	-	*12*	-	-	*19*	-
TIM4	-	-	*1*	*1*	*1*	-	*1*	*1*	*1*	-	*1*	-	*1*	-	-	*1*	-	-	-	-	-	*2*	*2*	-	*12*	-	-	-	-
TIM5	-	-	*1*	*1*	*1*	*1*	-	*1*	*1*	-	*1*	-	*1*	-	-	*1*	-	-	-	-	-	-	-	-	*12*	-	-	-	-
TIM6	*1*	*1*	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	-	*2*	*2*	-	*12*	-	-	*19*	*18*
TIM7	*1*	*1*	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	-	-	-	-	*12*	-	-	*19*	-
TIM8	*1*	*1*	*1*	*1*	*1*	*1*	*1*	-	*1*	-	*1*	-	*1*	-	-	*1*	-	-	-	-	-	*2*	*2*	-	*12*	-	-	*19*	-
TIM9	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	-	-	*1*	-	*1*	-	-	*1*	-	-	-	-	-	*2*	*2*	-	*12*	-	-	-	-
TIM10	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	-	-	-	-	-	-	*1*	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM11	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	-	-	-	-	-	-	*1*	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM12	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	-	-	-	*1*	-	-	*1*	-	-	-	-	-	*2*	*2*	-	*12*	-	-	-	-
TIM13	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	-	*1*	-	*1*	-	-	*1*	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM14	-	-	*1*	*1*	*1*	*1*	*1*	*1*	*1*	-	*1*	-	*1*	-	-	*1*	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM15	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	-	*1*	-	-	-	-	*1*	-	-	-	-	-	*2*	*2*	-	*12*	-	-	*19*	-
TIM16	-	-	*1*	*1*	*1*	*1*	*1*	*1*	*1*	-	*1*	-	*1*	-	-	*1*	-	-	-	-	-	-	*2*	-	-	-	-	-	-
TIM17	-	-	*1*	*1*	*1*	*1*	*1*	*1*	*1*	-	*1*	-	*1*	-	-	*1*	-	-	-	-	-	-	*2*	-	-	-	-	-	-
TIM20	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	-	*1*	-	*1*	-	-	-	-	-	-	-	-	*2*	*2*	-	*12*	-	-	-	-
LPTIM1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*2*	-	-	*12*	-	-	-	-

Table 81. Peripheral interconnect matrix (1) (continued)

Source	Destination																												
	HRTIM1	HRTIM2	TIM1	TIM2	TIM3	TIM4	TIM5	TIM8	TIM9	TIM10	TIM12	TIM13	TIM15	TIM16	TIM17	TIM20	LPTIM1	LPTIM2	LPTIM3	LPTIM4	LPTIM5	ADC1/2	ADC3/4	ADF	GP/HPDMA	TAMP	RTC	HSP	NPU
LPTIM2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	12	-	-	-	-
LPTIM3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	8	-	8	8	-	-	-	12	-	-	-	-
LPTIM4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	8	8	-	8	-	-	-	12	-	-	-	-
LPTIM5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	8	8	8	-	-	-	-	12	-	-	-	-
ADC1	3	3	3	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	14	-	-	-	-	-
ADC2	3	3	3	-	3	-	-	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	14	-	-	-	-	-
ADC3	3	3	-	-	-	3	-	3	-	-	-	-	-	-	-	3	-	-	-	-	-	-	-	-	-	-	-	-	-
ADC4	3	3	-	-	-	-	3	-	-	-	-	-	-	-	-	3	-	-	-	-	-	-	-	-	-	-	-	-	-
COMP1	4	4	4	4	4	4	4	4	-	-	-	-	4	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-
COMP2	4	4	4	4	4	4	4	4	-	-	-	-	4	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-
COMP3	4	4	4	4	4	4	4	4	-	-	-	-	4	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-
COMP4	4	4	4	4	4	4	4	4	-	-	-	-	4	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-
COMP5	4	4	4	4	4	4	4	4	-	-	-	-	4	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-
COMP6	4	4	4	4	4	4	4	4	-	-	-	-	4	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-
COMP7	4	4	4	4	4	4	4	4	-	-	-	-	4	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-
COMP8	4	4	4	4	4	4	4	4	-	-	-	-	4	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-
HSP	5	5	5	-	-	-	-	5	-	-	-	-	-	-	-	5	-	-	-	-	-	-	-	-	12	-	-	-	18
NPU	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	19	-
ADF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	19	-
GPDMA	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	12	-	-	19	18
HPDMA	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	12	-	-	19	18
U(S)ART	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LPUART	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	12	-	-	-	-
HSE / HSI / CSI	-	-	-	-	-	-	-	-	7	-	7	-	7	-	7	-	-	7	-	-	-	-	-	-	-	-	-	-	-
LSE / LSI	-	-	-	7	-	-	-	-	-	-	-	-	7	7	-	-	7	-	-	-	-	-	-	-	-	-	-	-	-
MCO1/2	-	-	-	-	-	-	-	-	7	-	7	-	7	-	7	-	-	-	-	-	-	-	-	-	-	-	-	-	-
I3C	-	-	-	-	-	-	-	-	-	17	-	17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SPI	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	12	-	-	-	-
EXTI	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	2	4	12	-	-	19	-
TAMP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	8	8	-	-	-	-	-	-	-	-	16	-	-
RTC Alarm	-	-	-	-	-	-	-	-	-	-	-	-	-	9	-	-	8	8	-	-	-	-	-	-	-	-	-	-	-
VBAT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13	-	-	-	15	-	-	-
VCORE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13	-	-	-	-	-	-
VREFINT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13	13	-	-	-	-	-	-
TEMP SENSE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13	-	-	-	15	-	-	-
CSS in LSE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	15	-	-

Table 81. Peripheral interconnect matrix (1) (continued)

Source	Destination																												
	HRTIM1	HRTIM2	TIM1	TIM2	TIM3	TIM4	TIM5	TIM8	TIM9	TIM10	TIM12	TIM13	TIM15	TIM16	TIM17	TIM20	LPTIM1	LPTIM2	LPTIM3	LPTIM4	LPTIM5	ADC1/2	ADC3/4	ADF	GP/HPDMA	TAMP	RTC	HSP	NPU
System error	*11*	*11*	*11*	-	-	-	-	*11*	-	-	-	-	*11*	*11*	*11*	*11*	-	-	-	-	-	-	-	-	-	-	-	-	-
USB OTG	-	-	-	*10*	*10*	*10*	*10*	-	*10*	-	*10*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

1. The “-” symbol in a gray cell means no interconnect.

11.3 Interconnection details

11.3.1 Master to slave interconnection for timers

From timer (TIM1/TIM2/TIM3/TIM4/TIM5/TIM8/TIM9/TIM12/TIM15/TIM16/TIM17/TIM20/HRTIM1/HRTIM2) to timer (TIM1/TIM2/TIM3/TIM4/TIM5/TIM8/TIM9/TIM12/TIM15/TIM16/TIM17/TIM20/HRTIM1/HRTIM2)

Purpose

The timers are linked together internally for timer synchronization or chaining.

When one timer is configured in master mode, it can reset, start, stop, or clock the counter of another timer configured in slave mode.

Timer synchronization

The synchronization modes are detailed in:

- *Section 34.3.35: Timer synchronization* for advanced-control timers TIM1/TIM8/TIM20
- *Section 35.4.27: Timer synchronization* for general-purpose timers TIM2/TIM3/TIM4/TIM5
- *Section 37.4.21: Timer synchronization (TIM9/TIM12 only)* for the general-purpose timers TIM9/TIM12
- *Section 38.4.33: Timer synchronization (TIM15 only)* for the general-purpose timer TIM15

Triggering signals

The output (from master) is on signal tim*x*_trgo for TIM1-9/12/15/20 as well as tim*x*_trgo2 for TIM1/8/20. In the case of TIM10/11/13/14/16/17, the output is on signal tim*x*_oc1. On HRTIM1/2, the output is hrtim*x*_out_sync2

The input (to slave) is on signals tim*x*_itr0-22 of TIM1-9/12/15/20, and hrtim*x*_eev1-16 of HRTIM1/2.

The possible master/slave connections are given in:

- *Table 373: Internal trigger connection* for advanced-control timers TIM1/TIM8/TIM20
- *Table 399: TIMx internal trigger connection* for the general-purpose timers TIM2/TIM3/TIM4/TIM5
- *Table 423: TIMx internal trigger connection* for general-purpose timers TIM9/TIM12
- *Table 435: TIMx internal trigger connection* for general-purpose timer TIM15
- Section 3.3.2: HRTIM pins and internal signals

Active power mode

Timers are optionally active in Run and Sleep modes. The effects of low-power modes on TIMx are given in:

- *Table 388: Effect of low-power modes on TIM1/TIM8/TIM20*
- *Table 416: Effect of low-power modes on TIM6/TIM7*
- *Table 409: Effect of low-power modes on TIM2/TIM3/TIM4/TIM5*
- *Table 425: Effect of low-power modes on TIM9/TIM10/TIM11/TIM12/TIM13/TIM14*
- *Table 446: Effect of low-power modes on TIM15/TIM16/TIM17*

11.3.2 Triggers to ADCs

From EXTI, timers TIM1/TIM2/TIM3/TIM4/TIM6/TIM9/TIM12/TIM15/TIM20, low-power timer LPTIM1, and HR timers HRTIM1/HRTIM2, to ADC1/2 and ADC3/4

Purpose

An analog to digital conversion, or sequence of conversions, can be triggered by a timer output capture or trigger, or an external event via an EXTI pin. For more details, see *Section 25.4.21: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)*.

Triggering signals

Triggers are on ADC inputs adc_ext0-31_trg for regular channels, and adc_jext0-31_trg for injected channels.

The trigger sources are:

- tim*x*_oc1-4 for TIM1-4/8/16/17/20
- tim*x*_trgo/trgo2 for TIM1-4/6-9/12/15/20
- hrtim*x*_adc_trg1-6 for HRTIM1/2
- lptim1_ch1/2 for LPTIM1
- EXTI11/EXTI2/EXTI15/EXTI3

See *Table 241: ADC1/2/3/4 interconnection* for details.

Active power mode

This interconnection is active in Run and Sleep modes for all ADCs. The timers are active in Run and Sleep mode only. The effects of low-power modes are given in:

- *Table 388: Effect of low-power modes on TIM1/TIM8/TIM20*
- *Table 409: Effect of low-power modes on TIM2/TIM3/TIM4/TIM5*
- *Table 425: Effect of low-power modes on TIM9/TIM10/TIM11/TIM12/TIM13/TIM14*

11.3.3 ADC analog watchdogs as triggers to timers

From ADC1/ADC2/ADC3/ADC4 to TIM1/TIM2/TIM3/TIM4/TIM5/TIM8/TIM20 and HRTIM1/HRTIM2

Purpose

The internal analog watchdog output signals coming from ADC1-4, are connected to on-chip timers. ADC1-4 can provide trigger event through analog watchdog signals to advanced-control timers TIM1-5/TIM8/TIM20 and to high-resolution timers HRTIM1/2 in order to reset, start, stop, or clock the counter.

For details on how to configure the external triggers on the timers, refer to:

- *Section 34.3.6: External trigger input* (TIM1/8/20)
- *Section 35.4.5: External trigger input* (TIM2/3/4/5)
- *Section 33.3.8: External events global conditioning* (HRTIM1/2)

For details of the analog watchdog configuration in the A-to-D converters, see:

- *Section 25.4.31: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)*

Triggering signals

The output from the ADC is on signals adc*x*_awd1-3. The input to timer TIM*x* is on signals tim*x*_etr17-22 (external trigger). The input to high-resolution timer HRTIM*x* is on signals hrtim*x*_eev1-16.

See *Table 374: Interconnect to the tim_etr input multiplexer* for the ADC analog watchdog connections to TIM1/8/20.

See *Table 400: Interconnect to the tim_etr input multiplexer* for the ADC analog watchdog connections to TIM2/3/4/5.

See *Table 326: HRTIM1/HRTIM2 external events mapping and associated features* for the ADC analog watchdog connections to HRTIM1/2.

Active power mode

ADC1-4 and timers are active in Run and Sleep modes.

11.3.4 Comparator outputs to timers

From COMP1-8 to TIM1/TIM2/TIM3/TIM4/TIM5/TIM8/TIM9/TIM10/TIM11/TIM12/TIM13/TIM14/TIM15/TIM16/TIM17/TIM20 and HRTIM1/HRTIM2

Purpose

The comparator output signals coming from COMP1-8 are connected to:

- Input capture inputs on the timers. This allows the timer counter value to be captured at the moment the comparator output transitions.
- Timer triggers. This allows the timer counter to be started or stopped on the comparator transitions.
- PWM timer TIM1/TIM8/TIM15/TIM16/TIM17/TIM20 break inputs and HRTIM1/HRTIM2 fault inputs, to halt their respective PWM outputs in a safe state in case a fault condition is detected by the comparators (for example, voltage/current out of range).

The comparator outputs programming is described in:

- *Section 29.4: COMP functional description*

The timer input capture programming is described in:

- *Section 34.3.9: Input capture mode* (TIM1/8/20)
- *Section 35.4.8: Input capture mode* (TIM2/3/4/5)
- *Section 37.4.7: Input capture mode* (TIM9-14)
- *Section 38.4.8: Input capture mode* (TIM15-17)
- *Section 33.3.4: Timer A..F timing units* (HRTIM1/2)

The timer external trigger configuration is described in:

- *Section 34.3.6: External trigger input* (TIM1/8/20)
- *Section 35.4.5: External trigger input* (TIM2/3/4/5)
- *Section 33.3.8: External events global conditioning* (HRTIM1/2)

The timer break/fault input configuration is described in:

- *Section 34.3.19: Using the break function* (TIM1/8/20)
- *Section 38.4.17: Using the break function* (TIM15-17)

Triggering signals

The comparator outputs are comp*x*_out.

The timer inputs are:

- tim*x*_ti1/2 for input capture
- tim*x*_etr for trigger
- tim*x*_break/break2 for PWM “break”

The high-resolution timer inputs are:

- hrtim*x*_eev1-16 for trigger
- hrtim*x*_in_flt1-8 for PWM “break”

See *Table 369: Interconnect to the tim_ti1 input multiplexer* for the comparator connections to TIM1/8/20 input capture.

See *Table 395: Interconnect to the tim_ti1 input multiplexer* and *Table 396: Interconnect to the tim_ti2 input multiplexer* for the comparator connections to TIM2/3/4/5 input capture.

See *Table 421: Interconnect to the tim_ti1 input multiplexer* and *Table 422: Interconnect to the tim_ti2 input multiplexer* for the comparator connections to TIM9-14 input capture.

See *Table 433: Interconnect to the tim_ti1 input multiplexer* and *Table 434: Interconnect to the tim_ti2 input multiplexer* for the comparator connections to TIM15-17 input capture.

See *Table 374: Interconnect to the tim_etr input multiplexer* for the comparator connections to TIM1/8/20 triggers.

See *Table 400: Interconnect to the tim_etr input multiplexer* for the comparator connections to TIM2/3/4/5 triggers.

See *Table 326: HRTIM1/HRTIM2 external events mapping and associated features* for the comparator connections to HRTIM1/2 input capture and triggers.

See *Table 375: Interconnect to the tim_brk_cmp multiplexer* and *Table 376: Timer break2 interconnect* for the comparator connections to TIM1/8/20 break inputs.

See *Table 436: Timer break interconnect* for the comparator connections to TIM15-17 break inputs.

Active power mode

COMP1-8 are active in Run, Sleep, and Stop (SVOS high) modes. Timers are active in Run and sleep modes.

11.3.5 HSP break to PWM timers

Break signals from HSP to PWM timers TIM1/TIM8/TIM15/TIM16/TIM17/TIM20 and HRTIM1/HRTIM2

Purpose

When the HSP stalls due to an error or in debug mode, the PWM timers can optionally be halted in safe state to avoid damage to the driver MOSFETs.

See *Section 34.3.19: Using the break function* for details of the break function in the PWM timers (TIM1/TIM8/TIM20).

See *Section 33.3.17: Fault protection* for details of the break function on the high-resolution timers (HRTIM1/HRTIM2). Note that the HSP break input is combined with the system fault input, and enabled via a programmable bit in the HRTIM option register.

See *Section 20.4.19: Break interface (BKITF)* for details of the HSP break feature.

Triggering signals

The HSP outputs are hsp_brk_out0/1/2/3. See *Table 139: HSP break connections on STM32H7Power* for the connections to the timers.

The PWM timer inputs are tim*x*_break/break2.

The high-resolution timer input is hrtim_sys_flt.

11.3.6 Triggers on ADF1

From EXTI to ADF1

Purpose

EXTI (EXTI15) can be used to generate a triggering event on ADF1 module and start an A-to-D conversion.

A description is given in:

- *Section 25.4.2: ADC pins and internal signals*
- *Section 30.4.9: Start-up sequence examples*

Triggering signals

The adf_trgi trigger input is the triggering input signal. The ADF trigger inputs connections are detailed in:

- *Table 294: ADF trigger connections*

Active power mode

This interconnection remains active down to Stop (SVOS high) for ADF1, assuming the trigger source remains active.

11.3.7 Internal input and clock sources to timers

From HSE, LSE, HSI, LSI, CSI, and MCO to TIM (TIM9/TIM12/TIM15/TIM16/TIM17) and LP timers (LPTIM1/LPTIM2/LPTIM3/LPTIM4/LPTIM5)

Purpose

A number of clock sources are connected to timer inputs to allow calibration of low accuracy sources against high accuracy clocks, and monitoring of external clock sources.

- HSI/1024 (HSI clock frequency divided by 1024) is connected to general-purpose timers TIM12 and TIM15 inputs, for calibrating the HSI oscillator frequency.
- HSE is connected to general-purpose timer TIM17 input, for monitoring the external crystal oscillator.
- LSE is connected to general-purpose timers TIM15 and TIM16 inputs, and as an external trigger on TIM2. This allows the external LSE crystal oscillator to be monitored, and also used to trigger a timer at the LSE frequency.
- LSI is connected to general-purpose timer TIM16 input, for calibrating the LSI oscillator frequency.
- The microcontroller output clock (MCO1/2) is connected as external input to general-purpose timers TIM9, TIM12, TIM15, and TIM17.
- LSI and LSE are connected to LPTIM1 inputs.
- HSI/1024 and CSI/128 are connected to LPTIM2 inputs.

Triggering signals

The TIM12/TIM15 inputs connected to the HSI/1024 are tim*x*_ti1_in4.

The HSE clock is connected to tim17_ti1_in2.

The LSE is connected to tim15_ti1_in10 and tim16_ti1_in3.

The LSI is connected to tim16_ti1_in2.

HSI/1024 is also connected to lptim2_ic2_mux1.

CSI/1024 is connected to lptim2_ic2_mux2.

LSE is connected to lptim1_ic2_mux2.

LSI is connected to lptim1_ic2_mux1.

See *Table 421: Interconnect to the tim_ti1 input multiplexer* for the TIM9/12 input connections.

See *Table 433: Interconnect to the tim_ti1 input multiplexer* for the TIM15/16/17 input connections.

See *Table 461: LPTIM1 input capture 2 connections* and *Table 462: LPTIM2 input capture 2 connections* for the LPTIM1/2 input connections.

Active power mode

This feature is available under Run and Sleep modes.

11.3.8 Triggers to low-power timers

From TAMP and RTC alarm to LP timers (LPTIM1/LPTIM2)

Purpose

LPTIM1/LPTIM2 counters may be started either by an active edge on one of the eight trigger inputs. This trigger feature is described in *Section 39.4.7: Trigger multiplexer* and the following sections.

Triggering signals

The input selection is described in *Table 457: LPTIM1/2/3/4/5 external trigger connections*.

Active power mode

This interconnection remains active down to Stop mode.

11.3.9 RTC wake-up as inputs to timers

From RTC to timers (TIM16)

Purpose

RTC wake-up interrupt can be used as input to general-purpose timer TIM16.

Triggering signals

RTC wake-up signal is connected to TIM16 tim_ti1_in4 signal as described in *Table 369: Interconnect to the tim_ti1 input multiplexer*.

Active power mode

This interconnection is active down to Stop. Timers are not active but the count is performed at wake-up.

11.3.10 OTG_FS SOF as trigger to timers

From OTG_HS to TIM2/TIM3/TIM4/TIM5/TIM9/TIM12

Purpose

OTG_HS SOF (start-of-frame) can generate triggers to the general-purpose timers TIM2/TIM3/TIM4/TIM5.

It can also generate triggers to the input capture inputs of TIM9/12.

This can be used, for example, to monitor and track audio packets received over the USB link.

Triggering signals

The otg_fs_sof is connected to tim_itr19 input of TIM2/TIM5, as well as tim_etr20 input of TIM2/TIM3/TIM4/TIM5, and the tim_ti1 inputs of TIM9/TIM12.

The connections to TIM2-5 are described in *Table 399: TIMx internal trigger connection* and *Table 400: Interconnect to the tim_etr input multiplexer*.

The connections to TIM9 and TIM12 are described in *Table 421: Interconnect to the tim_ti1 input multiplexer*.

Active power mode

This interconnection is active in Run and Sleep modes.

11.3.11 System errors as break signals to timers

The following system-level errors can trigger a break event in TIM1/TIM8/TIM15/TIM16/TIM17/TIM20 and HRTIM1/HRTIM2:

- Cortex-M7 lockup
- PVD low threshold
- CSS
- Double ECC errors in AXISRAM1-4, ITCM, DTCM, data cache, backup RAM, and flash memory.

Purpose

The purpose of the break event is to disable the timer outputs to protect the power transistors they control.

Triggering signals

The system level errors are connected to the tim_sys_brk0-5 inputs on TIM1/8/15-17/20. On HRTIM1/2, they are combined and connected to hrtim_sys_flt.

Active power mode

Timers are optionally active in Run and Sleep modes. System errors apply in RUN and Sleep modes.

11.3.12 Triggers to GPDMA/HPDMA

From EXTI, RTC, timers (TIM1/TIM2/TIM3/TIM4/TIM5/TIM6/TIM7/TIM9/TIM12/TIM15), LP timers (LPTIM1/LPTIM2/LPTIM3/LPTIM4/LPTIM5), HSP, HPDMA1/GPDMA1, SPI6, LPUART to HPDMA1/GPDMA1

Purpose

A HP/GPDMA trigger can be assigned to a HP/GPDMA channel x. A programmed HP/GPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event. The trigger mode can also be programmed to condition the LLI link transfer. More details are given in the sections below:

- *Section 13.3.6: GPDMA triggers* and *Section 12.3.6: HPDMA triggers*
- *Section 13.4.13: GPDMA triggered transfer* and *Section 12.4.13: HPDMA triggered transfer*

Triggering signals

The input triggers to the HPDMA and GPDMA are connected to the following signals:

- tim_trgo/trgo2 of TIM1/8/20
- tim_trgo of TIM2-7/9/12/15
- hsp1_trgo0-3 of HSP
- lptim_ch1/2 of LPTIM1-3
- lptim_out of LPTIM4/5
- exti0_wkup
- rtc_wkup
- lpuart_it
- spi6_it
- gpdma/hpdma_ch0-15_tc

GPDMA trigger mapping is specified in *Table 99: Programmed GPDMA1 trigger*, according to GPDMA_CxTR2.TRIGSEL[5:0].

HPDMA trigger mapping is specified in *Table 87: Programmed HPDMA1 trigger*, according to HPDMA_CxTR2.TRIGSEL[5:0].

Active power mode

Assuming sources are active in Stop modes, this interconnection remains functional in Stop.

Refer to:

- *Section 13.6: GPDMA in low-power modes*
- *Section 12.6: HPDMA in low-power modes*

11.3.13 Internal analog signals to analog peripherals

From internal analog source to ADC (ADC1/ADC2/ADC3/ADC4)

Purpose

The internal reference voltage (V_{REFINT}) is connected to ADC1/ADC2/ADC3/ADC4, mainly for safety reasons, to check if the ADC is still working. The internal temperature sensor (V_{SENSE}) monitoring channel is connected to ADC1. The internal digital core voltage (V_{CORE}) is connected to ADC3 for monitoring the core power supply. V_{BAT} is divided by four and connected to ADC1 for monitoring the back-up battery voltage. For more information refer to:

- *Section 25.4.35: Temperature sensor*
- *Section 25.4.36: VBAT supply monitoring*
- *Section 25.4.37: Monitoring the internal voltage reference*
- *Section 25.4.39: Monitoring the supply voltage*

Active power mode

These interconnections remain in Stop modes if the selected peripheral is kept active. Refer to *Section 25.5: ADC interrupts*.

11.3.14 ADC data filtering by the ADF1

From ADC1/ADC2 to ADF1

Purpose

ADC samples can be directly passed to the ADF for filtering, without the need for GP/HPDMA or software intervention.

See *Section 25.4.29: Managing conversions using the ADF*.

Triggering signals

See *Table 241: ADC1/2/3/4 interconnection*.

Active power mode

This feature is available in Run and Sleep mode.

11.3.15 Internal tamper sources

From internal peripherals, clocks or monitoring to tamper.

Purpose

Critical parameters, such as oscillator frequencies, power supply voltages, or on-chip temperature can be monitored and generate tamper events if out-of-specification values are detected. Tamper events can also be generated on system fault (for example, memory error) or unexpected debug activation.

A list of internal tamper sources can be found in *Table 134. TAMP interconnection*.

Active power mode

This interconnection is active in all power modes if the tamper source is activated.

11.3.16 Output from tamper to RTC

From TAMP to RTC

Purpose

The RTC can timestamp a tamper event in order to retrieve history in time of such detection. The RTC can also control GPIOs and set a signal based on tamp or alarm status outside the MCU.

Refer to *Section 43.3.3: GPIOs controlled by the RTC and TAMP* for more details.

Active power mode

This interconnection remains active in all power modes.

11.3.17 I3C in-band interrupt acknowledge to timers

From I3C to TIM10/TIM13

Purpose

When the I3C generates an in-band interrupt acknowledge, the timer counter value is captured in order to obtain a timestamp for I3C IBI sampled data.

Refer to *Section 45.9.14: I3C IBI transfer, as controller/target*.

Active power mode

This interconnection is active in Run and Sleep modes.

11.3.18 NPU input triggers

From TIM, HSP and HPDMA/GPDMA to NPU

Purpose

The NPU processing can be triggered by:

- TIM6 basic timer output trigger.
- HSP as part of a handshake with the NPU interrupt 3 output (npu_rdy).
- HPDMA or GPDMA channel 15 transfer complete trigger.

Triggering signals

TIM6 output trigger: tim6_trgo

HSP output triggers: hsp1_trgi5 - see *Table 138: HSP trigger output connections for STM32H7Power*.

HP/GPDMA transfer complete trigger: hp/gpdma1_ch15_tc

Active power mode

This interconnection is active in Run and Sleep modes.

11.3.19 HSP input triggers

Trigger signals from: Timers; NPU; ADF sound activity detector (SAD); GPDMA/HPDMA transfer complete (TC); EXTI0.

Purpose

The HSP processing can be triggered by the following events, in addition to software:

- Timer output triggers from TIM1/TIM2/TIM3/TIM6/TIM7/TIM8/TIM15
- NPU ready status on the npu_rdy signal (host interrupt #3). This signals to the HSP that the NPU is busy, or ready to read a batch of data from the HSP memory (SRAM3). The HSP acknowledges the latter by raising an hsp1_gpo output, upon which the NPU starts the transfer.
- ADF sound activity detection (SAD) signal, which goes active to prepare the HSP to receive and process audio samples. Refer to section *Section 30.4.4: ADC slave interface (ADCITF)*.
- HPDMA/GPDMA transfer complete events. These indicate that a transfer has completed on the associated DMA channel. If the DMA channel is used to transfer data to or from the HSP, the transfer complete event can trigger processing of the transferred data, or some other activity in the HSP.
- External event/interrupt. EXTI0 can be used to trigger HSP activity on an asynchronous external event.

The HSP input trigger behavior is described in *Section 20.4.9: Trigger input interface (TRGITF)*.

Triggering signals

See *Table 137: HSP trigger input connections for STM32H7Power* for the list of signals connected to HSP input triggers.

Active power mode

The HSP can be active in Run and Sleep modes.

12 High-performance direct memory access controller (HPDMA)

12.1 HPDMA introduction

The high-performance direct memory access (HPDMA) controller is a bus master and system peripheral.

The HPDMA is used to perform programmable data transfers between memory-mapped peripherals, and/or memories via linked-lists, upon the control of an off-loaded CPU.

12.2 HPDMA main features

- Single bidirectional 64-bit AXI master and single bidirectional 32-bit AHB master
- Memory-mapped data transfers from a source to a destination:
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
  - Peripheral-to-peripheral

  Transfer arbitration based on a 4-grade programmed priority at channel level:
  - One high-priority traffic class, for time-sensitive channels (queue 3)
  - Three low-priority traffic classes, with a weighted round-robin allocation for non time-sensitive channels (queues 0, 1, 2)
- Per channel event generation, on any of the following events: transfer complete, half-transfer complete, data transfer error, user setting error, link transfer error, completed suspension, and trigger overrun
- Per channel interrupt generation, with separately programmed interrupt enable per event
- 16 concurrent HPDMA channels:
  - Per channel FIFO for queuing source and destination transfers (see *Section 12.3.1*)
  - Intra-channel HPDMA transfers chaining via programmable linked-list into memory, supporting two execution modes: run-to-completion and link step mode
- Per linked-list item within a channel:
  - Separately programmed source and destination transfers
  - Programmable data handling between source and destination: byte-based reordering, packing or unpacking, padding or truncation, sign extension and left/right realignment
  - Programmable number of data bytes to be transferred from the source, defining the block level and byte-based padding
  - Linear source and destination addressing: either fixed or contiguously incremented addressing, programmed at block level, between successive burst transfers
  - 2D source and destination addressing: programmable signed address offsets between successive burst transfers (non-contiguous addressing within a block,

combined with programmable signed address offsets between successive blocks, at a second 2D/repeated block level, for a reduced set of channels (see *Section 12.3.1*)
  - Support for scatter-gather (multi-buffer transfers), data interleaving and deinterleaving via 2D addressing
  - Selection of programmable HPDMA request and trigger
  - Generation of programmable HPDMA half-transfer and transfer-complete event
  - Pointer to the next linked-list item and its data structure in memory, with automatic update of the HPDMA linked-list control registers
  - Channel abort and restart
  - Channel terminate with FIFO draining
- Debug:
  - Channel suspend and resume support
  - Channel status reporting, including FIFO level, and event flags
- Privileged/unprivileged support:
  - Support for privileged and unprivileged HPDMA transfers, independently at channel level
  - Privileged-aware AHB slave port

12.3 HPDMA implementation

12.3.1 HPDMA channels

A given HPDMA channel x is implemented with the following features and intended use.

To make the best use of the HPDMA performance, the following table lists some general recommendations, allowing the user to select and allocate a channel, given its implemented FIFO size and the requested HPDMA transfer.

Table 82. Implementation of HPDMA1 channels

Channel x	Hardware parameters		Features
	dma_fifo_size[x]	dma_addressing[x]	
x = 0 to 11	3	0	Channel x (x = 0 to 11) is implemented with: – a FIFO of 16 bytes, 4 words, 2 double-words – fixed/contiguously incremented addressing These channels can be used for HPDMA transfers between an APB or AHB peripheral, an AHB/AXI SRAM, or CPU TCM.
x = 12 to 15	5	1	Channel x (x = 12 to 15) is implemented with: – a FIFO of 64 bytes, 8 double-words – 2D addressing These channels can be also used for HPDMA transfers, including AXI external memories.

12.3.2 HPDMA in low-power modes

The HPDMA wake-up feature is implemented in the device low-power modes as per the table below.

Table 83. HPDMA1 in low-power modes

Feature	Low-power modes
Wake-up	HPDMA1 in Sleep mode

12.3.3 HPDMA requests

An HPDMA request from a peripheral can be assigned to a HPDMA channel x, via REQSEL[7:0] in HPDMA_CxTR2, provided that SWREQ = 0.

The HPDMA requests mapping is specified in the table below.

Table 84. Programmed HPDMA1 request

HPDMA_CxTR2.REQSEL[7:0]	Selected HPDMA request
0	adc1_dma
1	adc2_dma
2	adc3_dma
3	adc4_dma
4	dac1_ch1_dma
5	dac1_ch2_dma
6	dac2_ch1_dma
7	dac2_ch2_dma
8	dac3_ch1_dma
9	dac3_ch2_dma
10	dac4_ch1_dma
11	dac4_ch2_dma
12	hash_dma
13	hrtim1_master_dma
14	hrtim1_tima_dma
15	hrtim1_timb_dma
16	hrtim1_timc_dma
17	hrtim1_timd_dma
18	hrtim1_time_dma
19	hrtim1_timf_dma
20	hrtim2_master_dma
21	hrtim2_tima_dma
22	hrtim2_timb_dma

Table 84. Programmed HPDMA1 request (continued)

HPDMA_CxTR2.REQSEL[7:0]	Selected HPDMA request
23	hrtim2_timc_dma
24	hrtim2_timd_dma
25	hrtim2_time_dma
26	hrtim2_timf_dma
27	tim1_cc1_dma
28	tim1_cc2_dma
29	tim1_cc3_dma
30	tim1_cc4_dma
31	tim1_upd_dma
32	tim1_trgi_dma
33	tim1_com_dma
34	tim8_cc1_dma
35	tim8_cc2_dma
36	tim8_cc3_dma
37	tim8_cc4_dma
38	tim8_upd_dma
39	tim8_trgi_dma
40	tim8_com_dma
41	tim20_cc1_dma
42	tim20_cc2_dma
43	tim20_cc3_dma
44	tim20_cc4_dma
45	tim20_upd_dma
46	tim20_trgi_dma
47	tim20_com_dma
48	tim2_cc1_dma
49	tim2_cc2_dma
50	tim2_cc3_dma
51	tim2_cc4_dma
52	tim2_upd_dma
53	tim2_trgi_dma
54	tim3_cc1_dma
55	tim3_cc2_dma
56	tim3_cc3_dma
57	tim3_cc4_dma

Table 84. Programmed HPDMA1 request (continued)

HPDMA_CxTR2.REQSEL[7:0]	Selected HPDMA request
58	tim3_upd_dma
59	tim3_trgi_dma
60	tim4_cc1_dma
61	tim4_cc2_dma
62	tim4_cc3_dma
63	tim4_cc4_dma
64	tim4_upd_dma
65	tim4_trgi_dma
66	tim5_cc1_dma
67	tim5_cc2_dma
68	tim5_cc3_dma
69	tim5_cc4_dma
70	tim5_upd_dma
71	tim5_trgi_dma
72	tim6_upd_dma
73	tim7_upd_dma
74	tim15_cc1_dma
75	tim15_cc2_dma
76	tim15_upd_dma
77	tim15_trgi_dma
78	tim15_com_dma
79	tim16_cc1_dma
80	tim16_upd_dma
81	tim16_com_dma
82	tim17_cc1_dma
83	tim17_upd_dma
84	tim17_com_dma
85	spi1_rx_dma
86	spi1_tx_dma
87	spi2_rx_dma
88	spi2_tx_dma
89	spi3_rx_dma
90	spi3_tx_dma
91	spi4_rx_dma
92	spi4_tx_dma

Table 84. Programmed HPDMA1 request (continued)

HPDMA_CxTR2.REQSEL[7:0]	Selected HPDMA request
93	spi5_rx_dma
94	spi5_tx_dma
95	spi6_rx_dma
96	spi6_tx_dma
97	i2c1_rx_dma
98	i2c1_tx_dma
99	i2c2_rx_dma
100	i2c2_tx_dma
101	i2c3_rx_dma
102	i2c3_tx_dma
103	i3c1_rx_dma
104	i3c1_tx_dma
105	i3c1_tc_dma
106	i3c1_rs_dma
107	usart1_rx_dma
108	usart1_tx_dma
109	usart2_rx_dma
110	usart2_tx_dma
111	usart3_rx_dma
112	usart3_tx_dma
113	uart4_rx_dma
114	uart4_tx_dma
115	uart5_rx_dma
116	uart5_tx_dma
117	usart6_rx_dma
118	usart6_tx_dma
119	uart7_rx_dma
120	uart7_tx_dma
121	uart8_rx_dma
122	uart8_tx_dma
123	uart9_rx_dma
124	uart9_tx_dma
125	usart10_rx_dma
126	usart10_tx_dma
127	cordic_rd_dma

Table 84. Programmed HPDMA1 request (continued)

HPDMA_CxTR2.REQSEL[7:0]	Selected HPDMA request
128	cordic_wr_dma
129	fmac1_rd_dma
130	fmac1_wr_dma
131	lptim1_ic1_dma
132	lptim1_ic2_dma
133	lptim1_ue_dma
134	lptim2_ic1_dma
135	lptim2_ic2_dma
136	lptim2_ue_dma
137	lptim3_ic1_dma
138	lptim3_ic2_dma
139	lptim3_ue_dma
140	adf1_flt0_dma
141	usbpd_tx_dma
142	usbpd_rx_dma
143	lpuart1_rx_dma
144	lpuart1_tx_dma
145	xspi1_dma

12.3.4 HPDMA block requests

Some HPDMA requests must be programmed as a block request, and not as a burst request. Then BREQ in HPDMA_CxTR2 must be set for a correct HPDMA execution of the requested peripheral transfer at the hardware level.

Table 85. Programmed HPDMA1 request as a block request

HPDMA block requests
lptim1_ue_dma
lptim2_ue_dma
lptim3_ue_dma

12.3.5 HPDMA channels with peripheral early termination

An HPDMA channel, if implemented with this feature, can support the early termination of the data transfer from the peripheral which does also support this feature.

Table 86. HPDMA1 channel with peripheral early termination

HPDMA channel x with peripheral early termination
x = 0, x = 1, and x = 15

This HPDMA support is activated when the channel x is programmed with HPDMA_CxTR2.PFREQ = 1. Then, the peripheral itself can initiate and request a data transfer completion, before the HPDMA has transferred the whole block (see *Section 12.4.15* for more details).

12.3.6 HPDMA triggers

An HPDMA trigger can be assigned to an HPDMA channel x, via TRIGSEL[**5**:0] in HPDMA_CXTR2, provided that TRIGPOL[1:0] defines a rising or a falling edge of the selected trigger (TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

Table 87. Programmed HPDMA1 trigger

HPDMA_CxTR2.TRIGSEL[5:0]	Selected HPDMA trigger
0	tim1_trgo
1	tim1_trgo2
2	tim2_trgo
3	tim3_trgo
4	tim4_trgo
5	tim5_trgo
6	tim6_trgo
7	tim7_trgo
8	tim8_trgo
9	tim8_trgo2
10	tim9_trgo
11	tim12_trgo
12	tim15_trgo
13	tim20_trgo
14	tim20_trgo2
15	hsp1_trgo0
16	hsp1_trgo1
17	hsp1_trgo2
18	hsp1_trgo3
19	lptim1_ch1
20	lptim1_ch2
21	lptim2_ch1
22	lptim2_ch2

Table 87. Programmed HPDMA1 trigger (continued)

HPDMA_CxTR2.TRIGSEL[5:0]	Selected HPDMA trigger
23	lptim3_ch1
24	lptim3_ch2
25	lptim4_out
26	lptim5_out
27	exti_exti0_wkup
28	exti_wkup_rtc_wkup
29	lpuart1_it_r_wup_async
30	lpuart1_it_t_wup_async
31	spi6_it
32	gpdma1_ch0_tc
33	gpdma1_ch1_tc
34	gpdma1_ch2_tc
35	gpdma1_ch3_tc
36	gpdma1_ch4_tc
37	gpdma1_ch5_tc
38	gpdma1_ch6_tc
39	gpdma1_ch7_tc
40	gpdma1_ch8_tc
41	gpdma1_ch9_tc
42	gpdma1_ch10_tc
43	gpdma1_ch11_tc
44	gpdma1_ch12_tc
45	gpdma1_ch13_tc
46	gpdma1_ch14_tc
47	gpdma1_ch15_tc
48	hpdma1_ch0_tc
49	hpdma1_ch1_tc
50	hpdma1_ch2_tc
51	hpdma1_ch3_tc
52	hpdma1_ch4_tc
53	hpdma1_ch5_tc
54	hpdma1_ch6_tc
55	hpdma1_ch7_tc
56	hpdma1_ch8_tc
57	hpdma1_ch9_tc

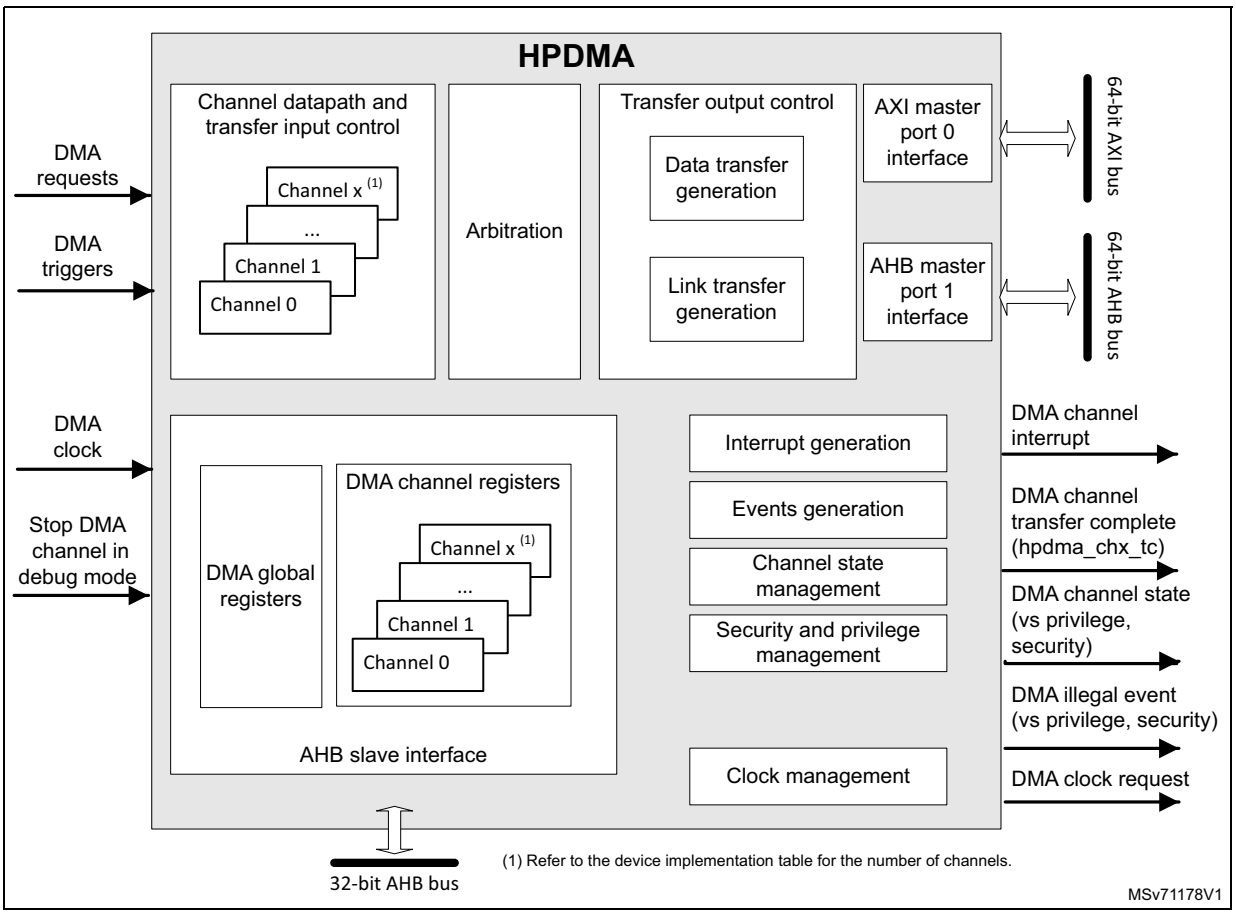
Table 87. Programmed HPDMA1 trigger (continued)

HPDMA_CxTR2.TRIGSEL[5:0]	Selected HPDMA trigger
58	hpdma1_ch10_tc
59	hpdma1_ch11_tc
60	hpdma1_ch12_tc
61	hpdma1_ch13_tc
62	hpdma1_ch14_tc
63	hpdma1_ch15_tc

12.4 HPDMA functional description

12.4.1 HPDMA block diagram

Figure 79. HPDMA block diagram



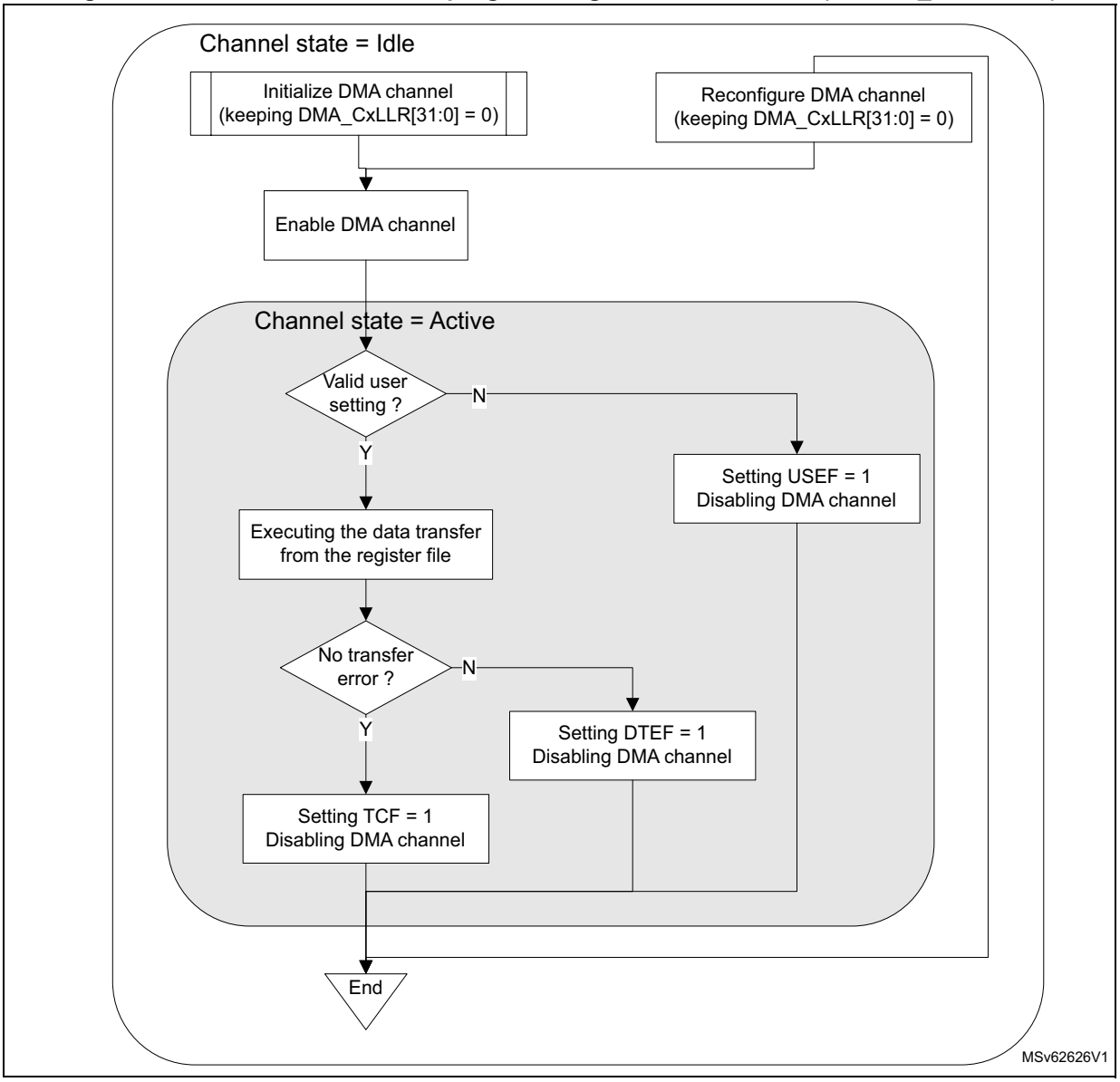
12.4.2 HPDMA channel state and direct programming without any linked-list

After an HPDMA reset, an HPDMA channel x is in idle state. When the software writes 1 into the HPDMA_CxCR.EN enable control bit, the channel takes into account the value of the different channel configuration registers (HPDMA_CxXXX), switches to the active/non-idle state, and starts to execute the corresponding requested data transfers.

After enabling/starting an HPDMA channel transfer by writing 1 into HPDMA_CxCR.EN, an HPDMA channel interrupt on a complete transfer notifies the software that the HPDMA channel is back in idle state (EN is then deasserted by hardware), and that the channel is ready to be reconfigured then enabled again.

The figure below illustrates this HPDMA direct programming without any linked-list (HPDMA_CxLLR = 0).

Figure 80. HPDMA channel direct programming without linked-list (HPDMA_CxLLR = 0)



12.4.3 HPDMA channel suspend and resume

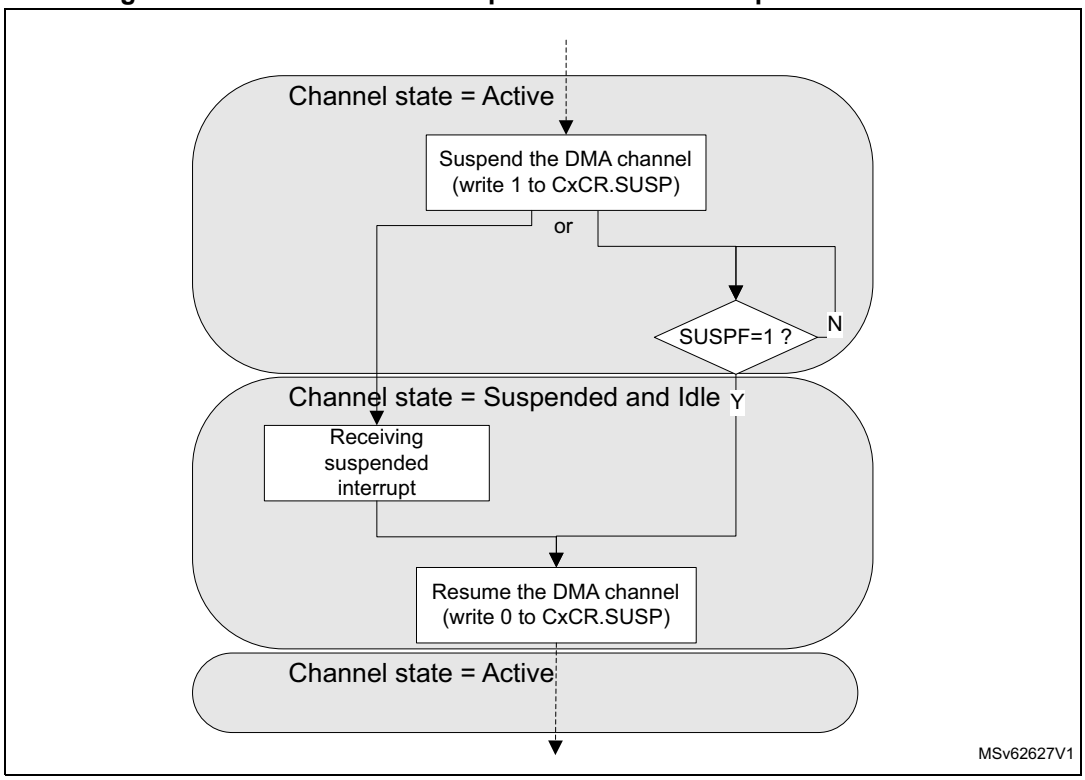
The software can suspend on its own a channel still active, with the following sequence:

1. The software writes 1 into the HPDMA_CxCR.SUSP bit.
2. The software polls the suspended flag HPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to HPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing HPDMA transfer over its master ports. Then the software can observe, in a steady state, any read register or bitfield that is hardware modifiable.

Note: An ongoing HPDMA transfer can be a data transfer (a source/destination burst transfer) or a link transfer for the internal update of the linked-list register file from the next linked-list item.

3. The software safely resumes the suspended channel by writing 0 to HPDMACxCR.SUSP.

Figure 81. HPDMA channel suspend and resume sequence



Note: A suspend and resume sequence does not impact the HPDMA_*CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.*

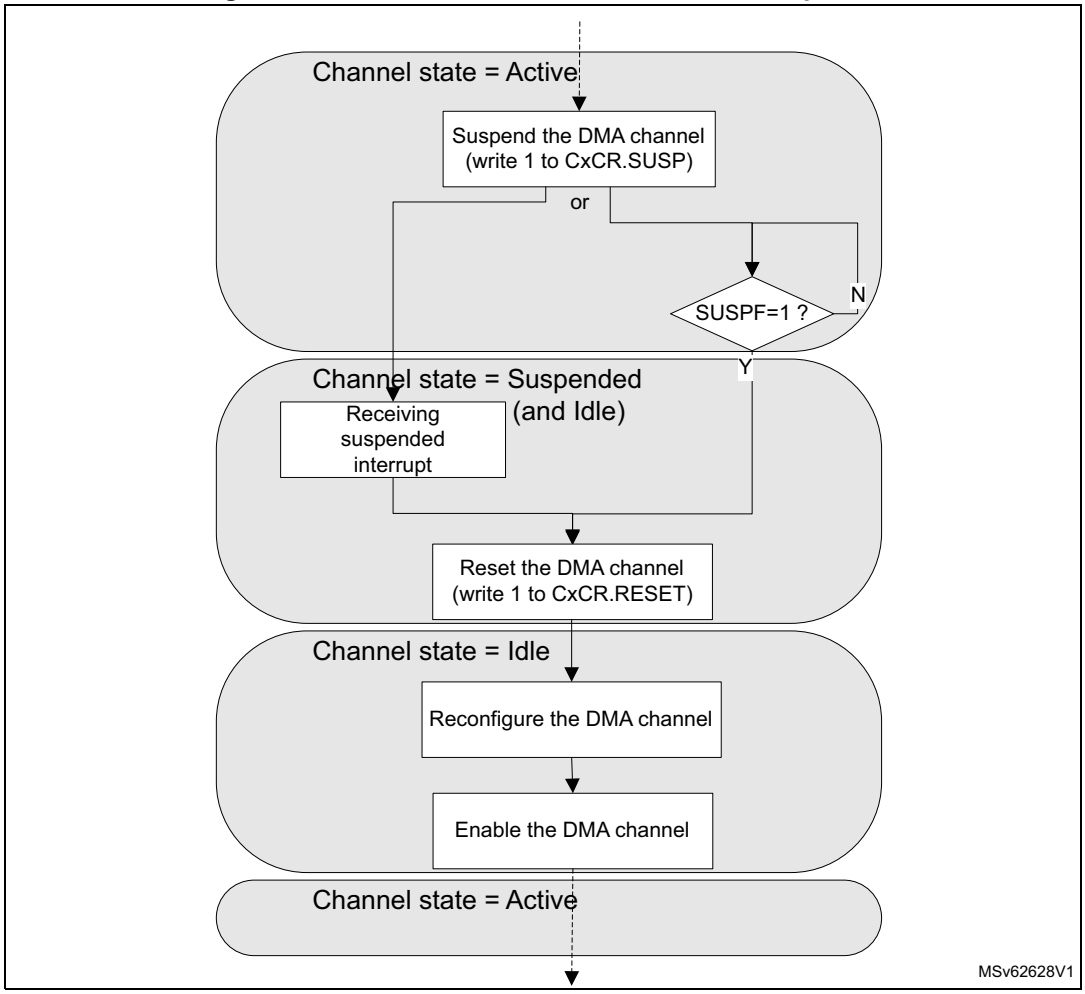
12.4.4 HPDMA channel abort and restart

Alternatively, like for aborting a continuous HPDMA transfer with a circular buffering or a double buffering, the software can abort, on its own, a still active channel with the following sequence:

1. The software writes 1 into the HPDMA_CxCR.SUSP bit.
2. The software polls suspended flag HPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to HPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing HPDMA transfer over its master port.
3. The software resets the channel by writing 1 to HPDMA_CxCR.RESET. This causes the reset of the FIFO, the reset of the channel internal state, the reset of the HPDMA_CxCR.EN bit, and the reset of the HPDMA_CxCR.SUSP bit.
4. The software safely reconfigures the channel. The software must reprogram hardware-modified HPDMA_CxBR1, HPDMA_CxSAR, and HPDMA_CxDAR.

5. In order to restart the aborted then reprogrammed channel, the software enables it again by writing 1 to the HPDMA_CxCR.EN bit.

Figure 82. HPDMA channel abort and restart sequence



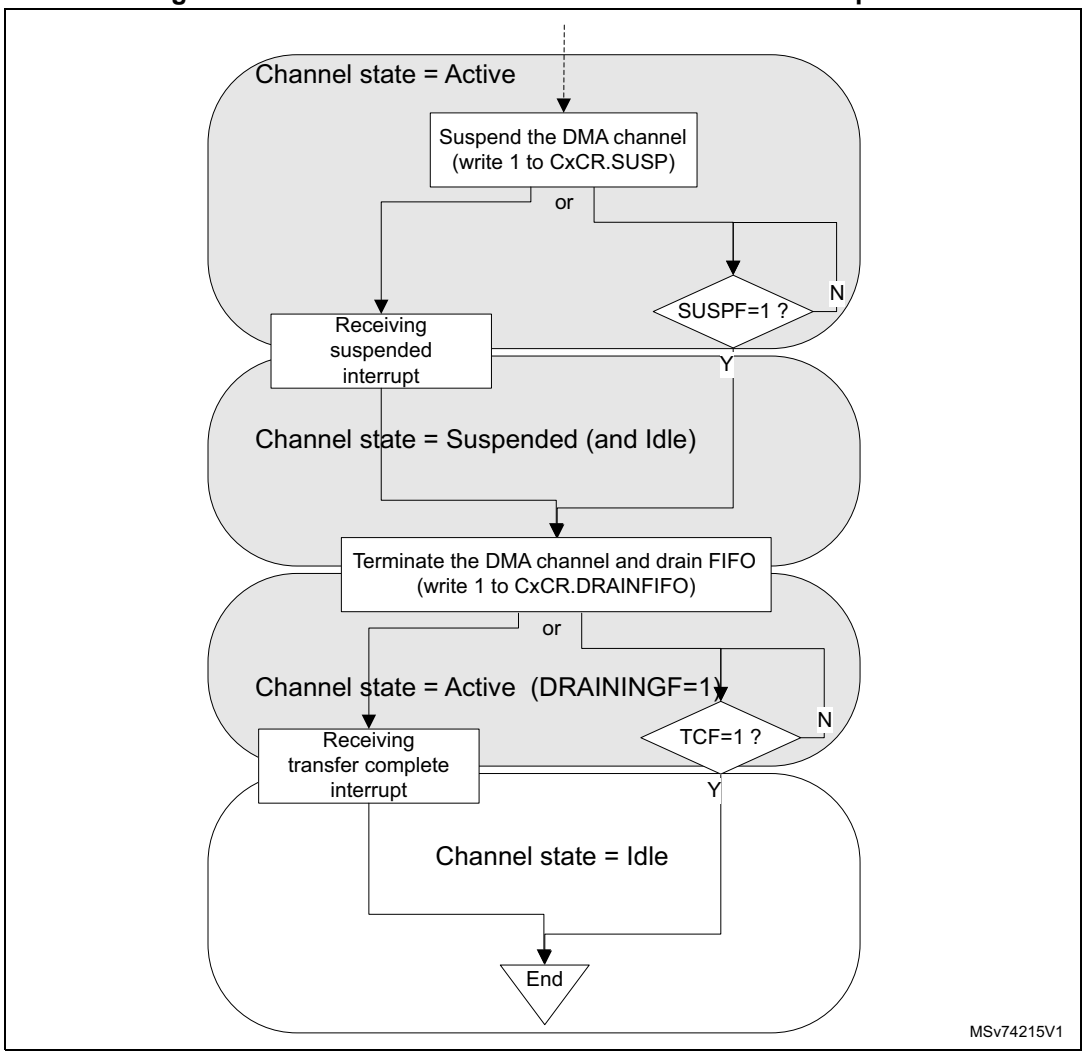
12.4.5 HPDMA channel terminate and drain FIFO

Alternatively, for terminating a continuous HPDMA transfer with a circular buffering or a double buffering, and having popped out any residual byte(s) in the channel FIFO and writing them to the destination, the software can terminate an active channel and drain the channel FIFO with the following sequence:

1. The software writes 1 into the HPDMA_CxCR.SUSP bit.
2. The software polls suspended flag HPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to HPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing HPDMA transfer over its master port.
3. The software orders to drain the channel FIFO by writing 1 to HPDMA_CxCR.DRAINFIFO.
4. The software polls transfer complete flag HPDMA_CxSR.TCF until TCF = 1, or waits for an interrupt previously enabled by writing 1 to HPDMA_CxCR.TCIE. Wait for the

channel to be effectively with a drained FIFO means wait for completion of any residual write burst from the FIFO to the destination. Then the channel is disabled by hardware and in IDLE state.

Figure 83. HPDMA channel terminate and drain FIFO sequence



Note: If the softwares writes the HPDMA_CxCR register in a single shot with concurrently SUSP = 1, DRAINFIFO = 1, and RESET = 0, then the sequence for terminating and draining the FIFO also occurs.

HPDMA_CxTR2.DREQ = 1 and HPDMA_CxCR.DRAINFIFO = 1 is not supported, only peripheral-to-memory and memory-to-memory transfers are supported to be terminated with drained FIFO.

12.4.6 HPDMA linked-list data structure

Alternatively to the direct programming mode, a channel can be programmed by a list of transfers, known as a list of linked-list items (LLI). Each LLI is defined by its data structure.

For a channel x, the base address in memory of the data structure of a next LLI_{n+1} is the sum of the following:

- The link base address of the channel x (in HPDMA_CxLBAR)
- The link address offset LA[15:2] bitfield in HPDMA_CxLLR, that is the updated result from the data structure of the previous LLIn of the channel x

The data structure for each LLI can be specific.

A linked-list data structure is addressed following the value of UT1, UT2, UB1, USA, UDA, and ULL bits, plus UB2 and UT3 when present, in HPDMA_CxLLR.

In linked-list mode, each HPDMA linked-list register (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR or HPDMA_CxLLR, plus HPDMA_CxTR3 or HPDMA_CxBR2 when present) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of HPDMA_CxLLR that was conditionally updated from the linked-list data structure of the previous LLI.

Caution: The user must program the pointer to the next linked-list data structure (HPDMA_CxLLR[15:0]) not to exceed the 64-Kbyte addressable space defined by the link base address register (HPDMA_CxLBAR). The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Static linked-list data structure

For example, when the update bits (UT1, UT2, UB1, USA, UDA, and ULL, plus UB2 and UT3 when present) in HPDMA_CxLLR are all asserted, the linked-list data structure in the memory is maximal with:

- For channels without 2D addressing capability: Six contiguous 32-bit locations, including HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR (see *Figure 84*) and including the first linked-list register file (LLI_0), and the next LLIs (LLI_1, LLI_2,...) in the memory
- For channels with 2D addressing capability: eight contiguous 32-bit locations, including HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR, plus HPDMA_CxTR3 and HPDMA_CxBR2 (see *Figure 85*), including the first linked-list register file (LLI_0), and the next LLIs (LLI_1, LLI_2,...) in the memory

Figure 84. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x

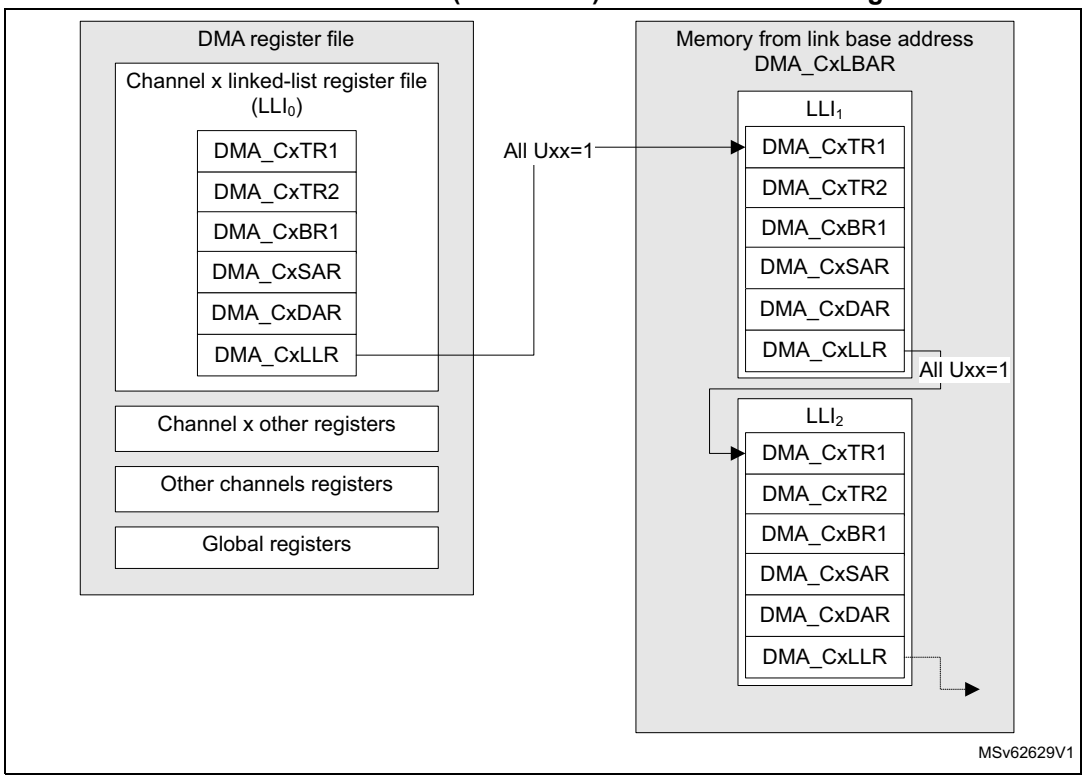
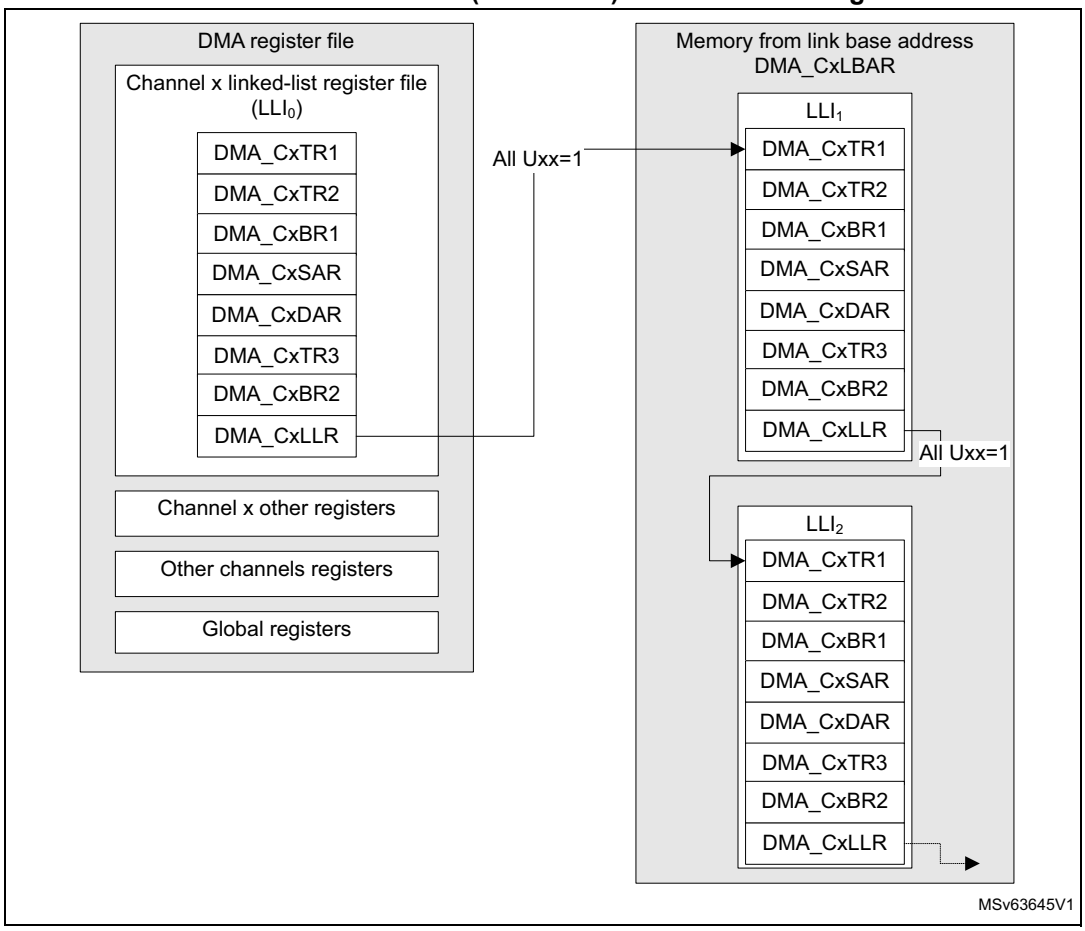


Figure 85. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x



Dynamic linked-list data structure

Alternatively, the memory organization for the full list of LLIs can be compacted with specific data structure for each LLI.

If UT1 = 0 and UT2 = 1, the link address offset of HPDMA_CxLLR points to the updated value of HPDMA_CxTR2, instead of HPDMA_CxTR1 which is not to be modified (see *Figure 86*).

Example: if UT1 = UB1 = USA = 0, and if UT3 = UB2 = 0 when channel x is with 2D addressing, and if UT2 = UDA = ULL = 1, the next LLI does not contain an (updated) value for HPDMA_CxTR1, nor HPDMA_CxBR1, nor HPDMA_CxSAR, nor HPDMA_CxTR3, nor HPDMA_CxBR2 when channel x is with 2D addressing. The next LLI contains an updated value for HPDMA_CxTR2, HPDMA_CxDAR, and HPDMA_CxLLR, as shown in *Figure 87*.

Figure 86. HPDMA dynamic linked-list data structure of linear addressing channel x

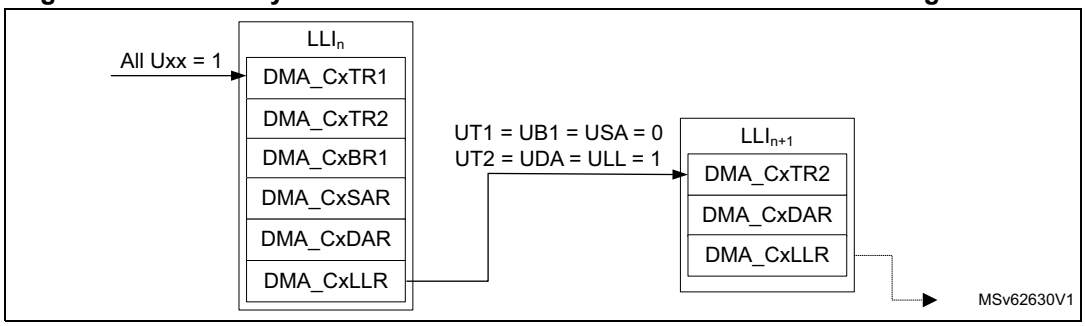
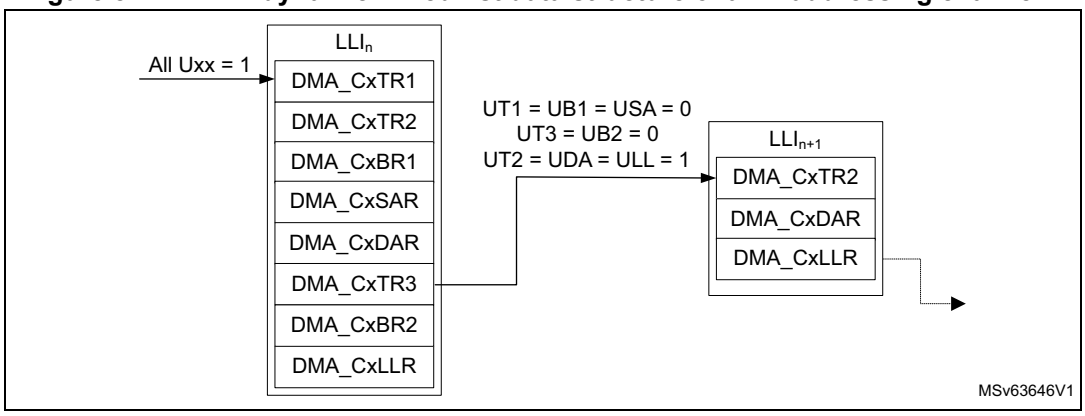


Figure 87. HPDMA dynamic linked-list data structure of a 2D addressing channel x



12.4.7 Linked-list item transfer execution

An LLI_n transfer is the sequence of:

1. A data transfer: the HPDMA executes the data transfer as described by the HPDMA internal register file (this data transfer can be void/null for LLI_0).
2. A conditional link transfer: the HPDMA automatically and conditionally updates its internal register file by the data structure of the next LLI_{n+1}, as defined by the HPDMA_CxLLR value of the LLI_n.

Note: The initial data transfer, as defined by the internal register file (LLI_0), can be null (HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxTR2.PFREQ = 0), provided that UB1 is set in HPDMA_CxLLR (meaning there is a non-null data transfer described by the next LLI_1 in the memory to be executed).

Depending on the intended HPDMA use, an HPDMA channel x can be executed as described by the full linked-list (run-to-completion mode, HPDMA_CxCR.LSM = 0), or can be programmed for a single execution of a LLI (link step mode, HPDMA_CxCR.LSM = 1), as described in the next sections.

12.4.8 HPDMA channel state and linked-list programming in run-to-completion mode

When HPDMA_CxCR.LSM = 0 (in full-list execution mode, execution of the full sequence of LLIs, named run-to-completion mode), an HPDMA channel x is initially programmed, started

by writing 1 to HPDMA_CxCR.EN, and after completed at channel level.
The channel transfer is:

- Configured with at least the following:
  - The first LLI_0, internal linked-list register file: HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR, plus HPDMA_CxTR3 and HPDMA_CxBR2 when present
  - The last LLI_N described by the linked-list data structure in memory, as defined by HPDMA_CxLLR reflecting the before last LLI_{N-1}
- Completed when HPDMA_CxLLR[31:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0, if BRC[10:0] is present and HPDMA_CxBR1.BNDT[15:0] = 0, at the end of the last LLI_{N-1} transfer

HPDMA_CxLLR[31:0] = 0 is the condition of a linked-list based channel completion, and means the following:

- The 16 low significant bits HPDMA_CxLLR.LA[15:0] of the next link address are null.
- All bits HPDMA_CxLLR.Uxx are null (UT1, UT2, UB1, USA, UDA, and ULL, plus UB2 and UT3 when present).

The channel may never be completed when HPDMA_CxLLR.LSM = 0:

- If the last LLI_N is recursive, pointing to itself as a next LLI, in one of the following:
  - HPDMA_CxLLR.ULL = 1 and HPDMA_CxLLR.LA[15:2] is updated with the same value.
  - HPDMA_CxLLR.ULL = 0
- If LLI_N points to a previous LLI

At the regular data transfer completion at a block level, HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0 (if present). Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 12.4.15* for more details).

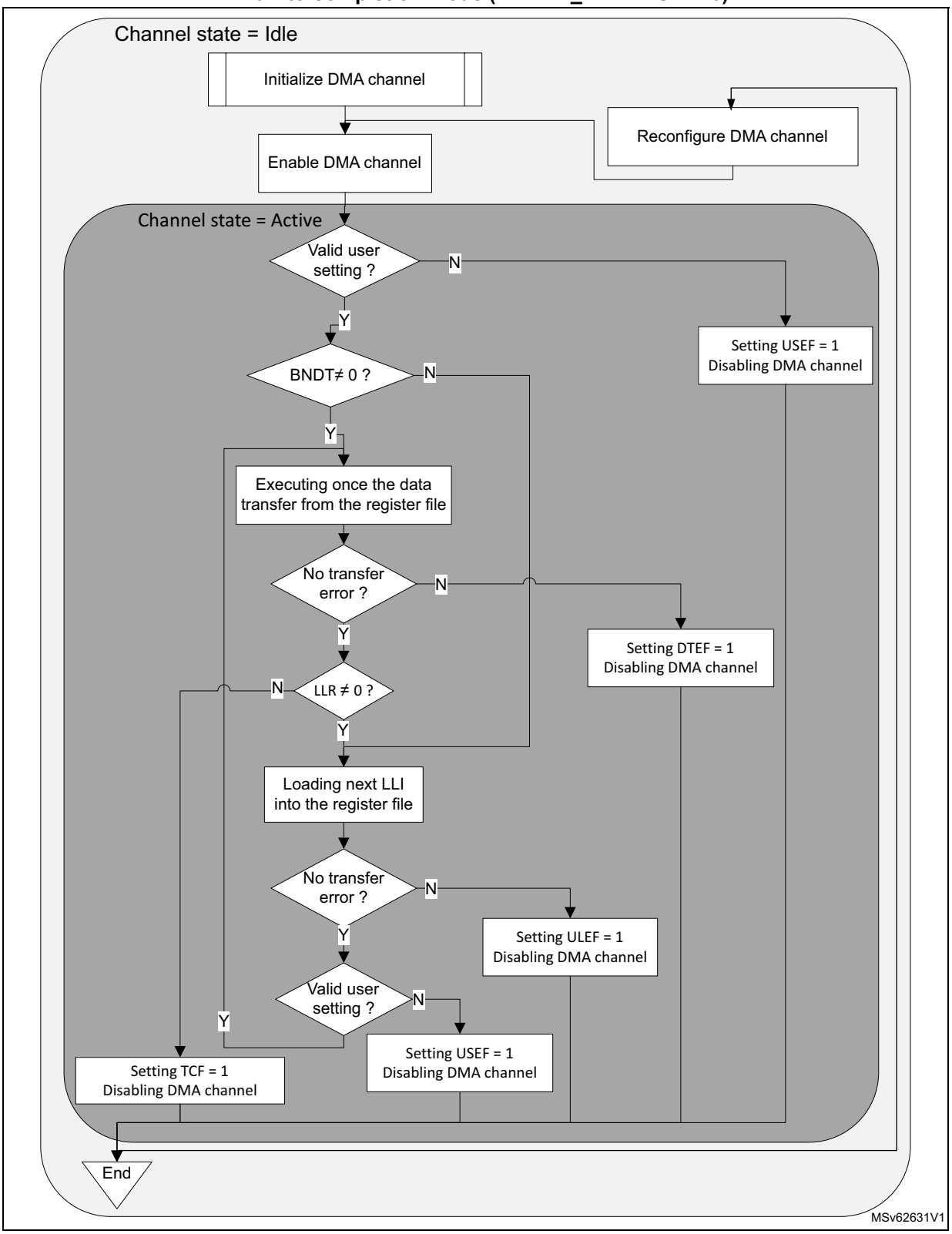
In typical run-to-completion mode, allocation of an HPDMA channel, including detailed programming, occurs once during HPDMA initialization. This process reserves a data communication link and HPDMA service during run-time for repeated transfers, such as transfers from a peripheral to memory, from memory to a peripheral, or memory-to-memory transfers. The reserved data communication link may consist of a dedicated channel, or the channel may be shared. A repeated transfer consists of a sequence of linked list items (LLIs).

Figure 88 depicts the HPDMA channel execution and its registers programming in run-to-completion mode.

Note: Figure 88 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in HPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at channel completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion, the hardware always set TCEF = 1 and disables the channel.

In Figure 88, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 12.3.5).

Figure 88. HPDMA channel execution and linked-list programming in run-to-completion mode (HPDMA_CxCR.LSM = 0)



Run-time inserting a LLI_n via an auxiliary channel, in run-to-completion mode

The start of the link transfer of the LLI_{n-1} (start of the LLI_n loading) can be conditioned by the occurrence of a trigger, when programming the following bitfields of HPDMA_CxTR2 in the data structure of the LLI_{n-1}:

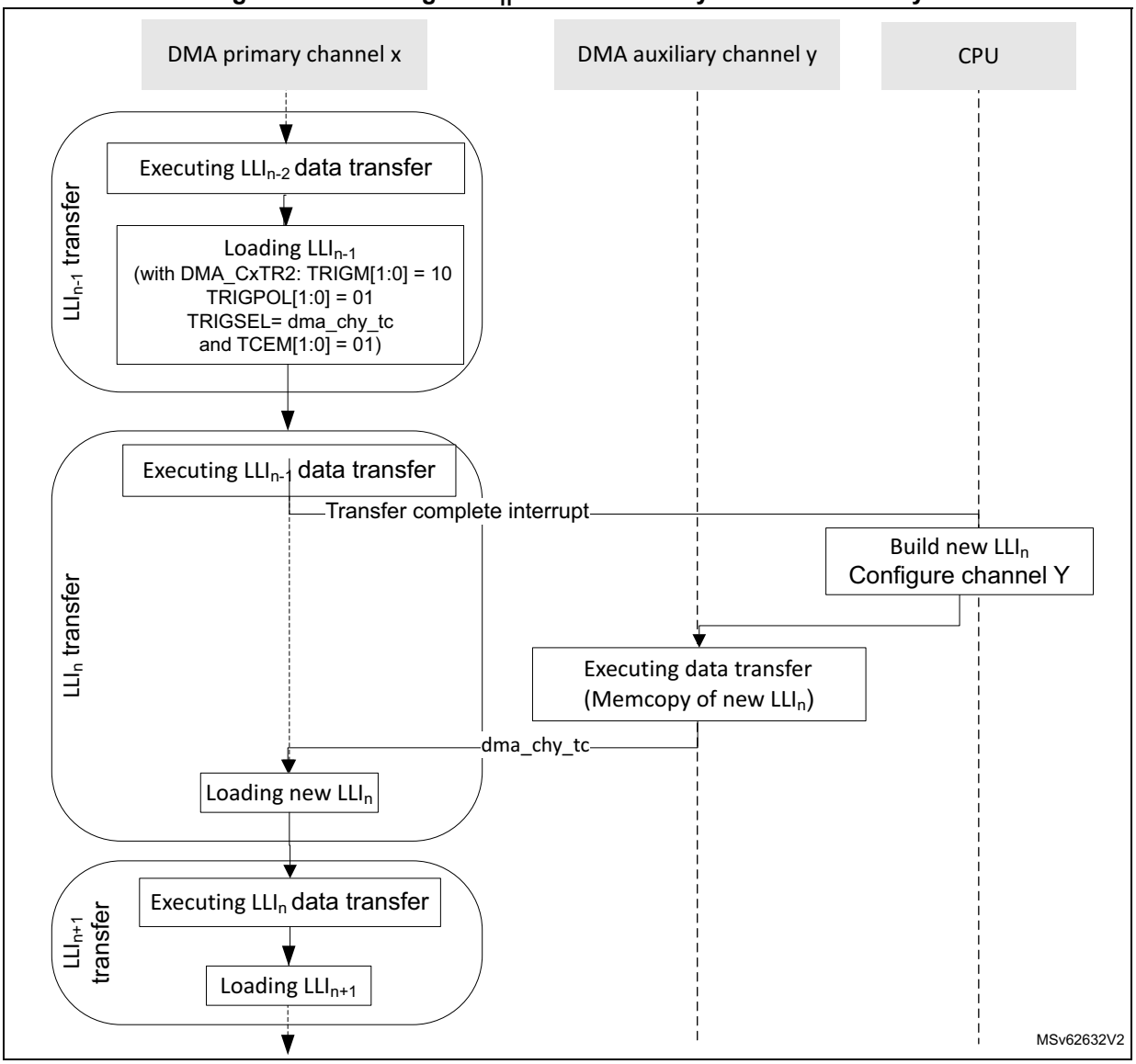
- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[5:0] (see *Section 12.3.6* for the trigger selection details)

Another auxiliary channel y can be used to store the channel x LLI_n in the memory, and to generate a transfer complete event hpdma_chy_tc. By selecting this event as the input trigger of the link transfer of the LLI_{n-1} of the channel x, the software can pause the primary channel x after its LLI_{n-1} data transfer, until it is indeed written the LLI_n.

Figure 89 depicts such a dynamic elaboration of a linked-list of a primary channel x, via another auxiliary channel y.

Caution: This use case is restricted to an application with an LLI_{n-1} data transfer that does not need a trigger. The triggering mode of this LLI_{n-1} is used to load the next LLI_n.

Figure 89. Inserting a LLI_n with an auxiliary HPDMA channel y



12.4.9 HPDMA channel state and linked-list programming in link step mode

When HPDMA_CxCR.LSM = 1 (in link step execution mode, single execution of one LLI), a channel transfer is executed and completed after each single execution of a LLI, including its (conditional) data transfer and its (conditional) link transfer.

An HPDMA channel transfer can be programmed at an LLI level, started by writing 1 into HPDMA_CxCR.EN, and after completed at LLI level:

- The current LLI_n transfer is described with the following:
  - HPDMA_CxTR1 defines the source/destination elementary single/burst transfers.
  - HPDMA_CxBR1 defines the number of bytes at a block level (BNDT[15:0]), and for channels with 2D addressing it defines also the number of blocks at a

2D/repeated block level (BRC[10:0]+1), and the incrementing/decrementing mode for address offsets.

  - HPDMA_CxTR2 defines the input control (request, trigger), and the output control (transfer-complete event) of the transfer.
  - HPDMA_CxSAR and HPDMA_CxDAR define the source/destination transfer start address.
  - HPDMA_CxTR3 for channels with 2D addressing defines the source/destination additional address offset between burst transfers.
  - HPDMA_CxBR2 for channels with 2D addressing defines the source/destination additional address offset between blocks at a 2D/repeated block level.
  - HPDMA_CxLLR defines the data structure, and the address offset of the next LLI_{n+1} in the memory.
- The current LLI_n transfer is completed after the single execution of the current LLI_n:
  - After the (conditional) data transfer completion (when HPDMA_CxBR1.BRC[10:0] = 0 if BRC[10:0] is present, and HPDMA_CxBR1.BNDT[15:0] = 0)
  - After the (conditional) update of the HPDMA link register file from the data structure of the next LLI_{n+1} in memory

Note: If an LLI is recursive (pointing to itself as a next LLI, either HPDMA_CxLLR.ULL = 1 and HPDMA_CxLLR.LA[15:2] is updated with the same value, or HPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

At the regular data transfer completion at a block level, HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0 (if present). Alternatively, a block transfer can be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 12.4.15* for more details).

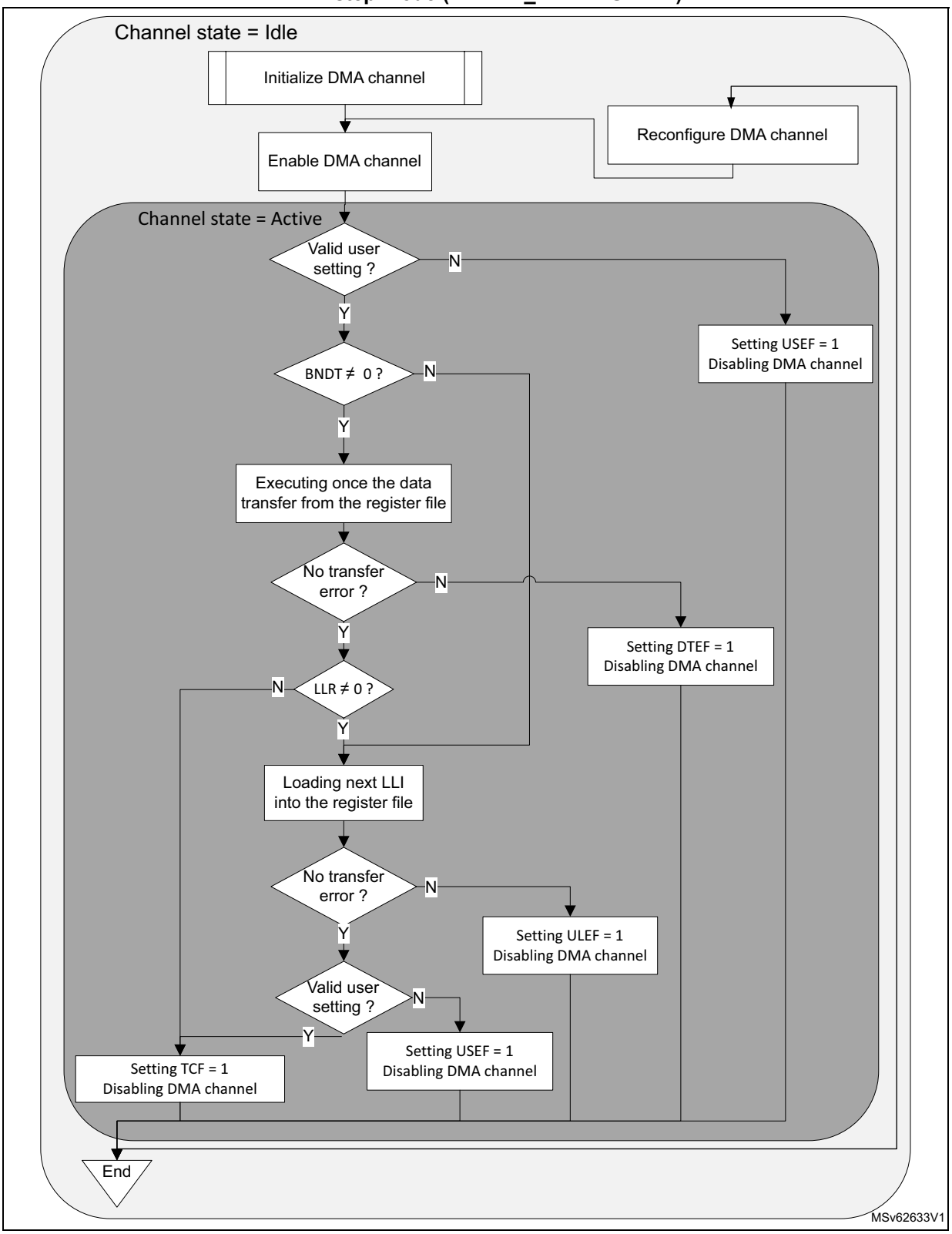
The link step mode can be used to elaborate dynamically LLIs in memory during run-time. The software can be facilitated by using a static data structure for any LLI_n (all update bits of HPDMA_CxLLR have a static value, LLI_n.LLR.LA = LLI_{n-1}.LLR.LA + constant).

Figure 90 depicts the HPDMA channel execution mode, and its programming in link step mode.

Note: Figure 90 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in HPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 90, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes BNDT ≠ 0 and PFREQ = 1 if the peripheral requests a data transfer with early termination (see Section 12.3.5).

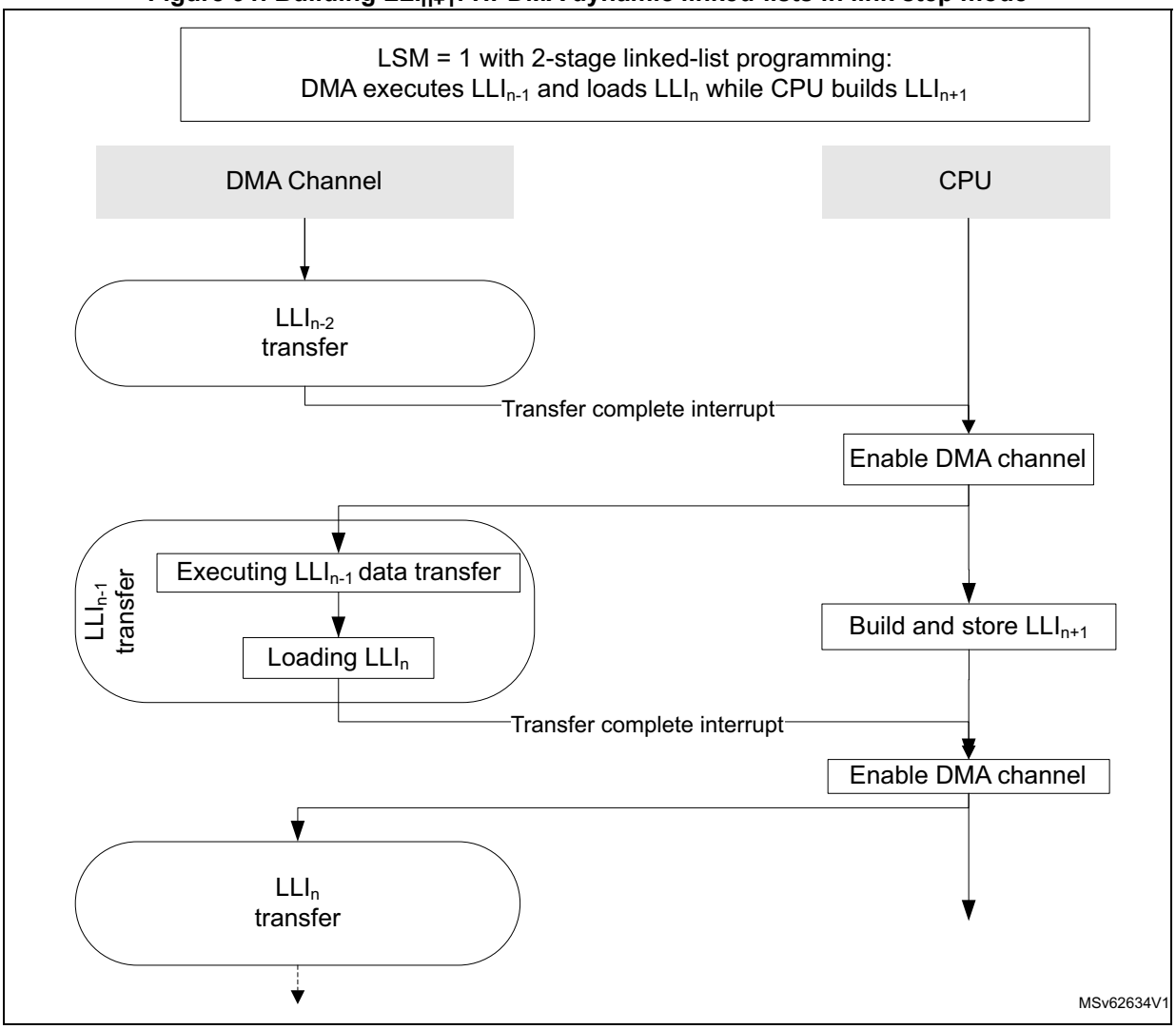
Figure 90. HPDMA channel execution and linked-list programming in link step mode (HPDMA_CxCR.LSM = 1)



Run-time adding a LLI_{n+1} in link step mode

During run-time, the software can defer the elaboration of the LLI_{n+1} (and next LLIs), until/after the HPDMA executed the transfer from the LLI_{n-1} and loaded the LLI_n from the memory, as shown in the figure below.

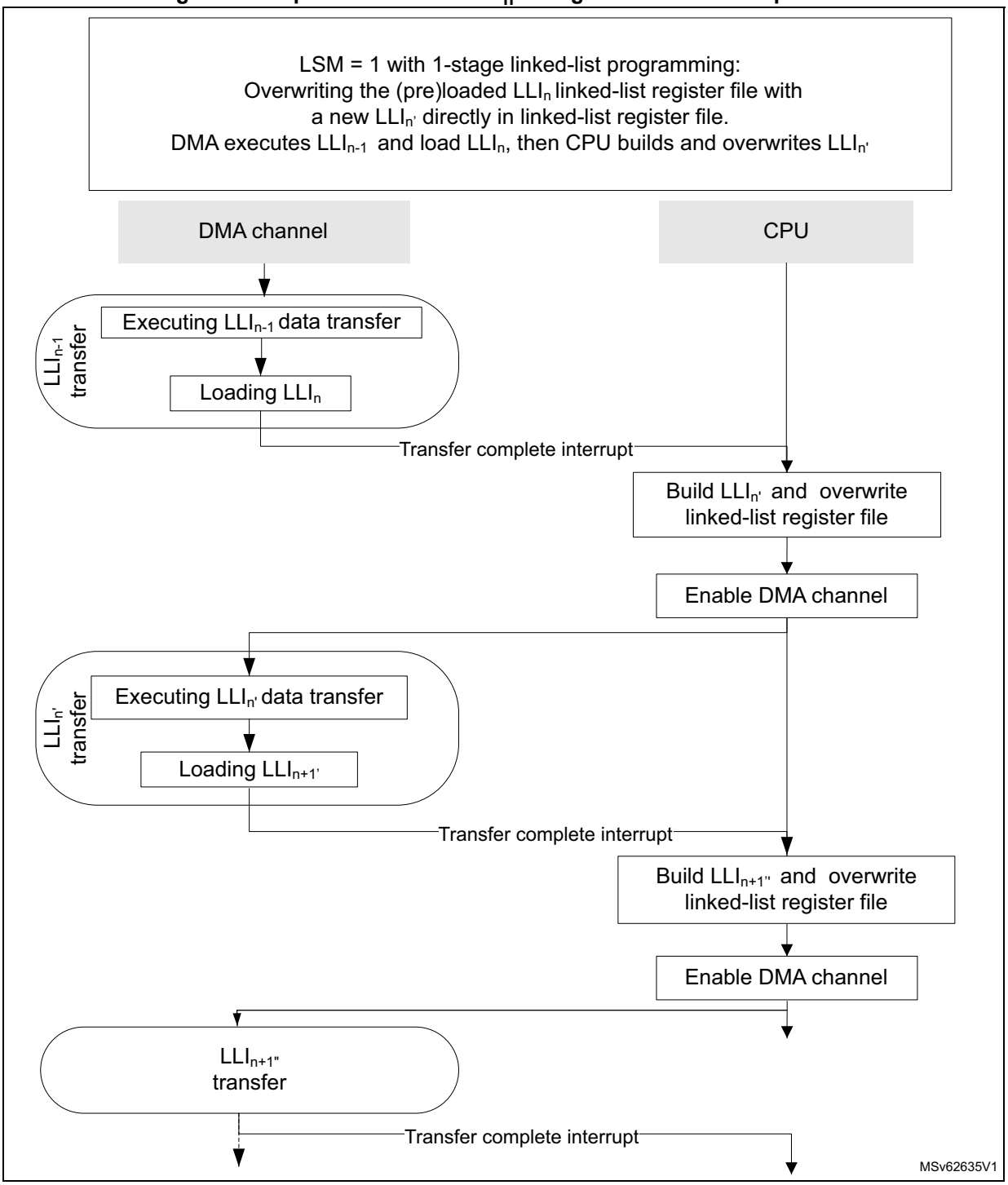
Figure 91. Building LLI_{n+1}: HPDMA dynamic linked-lists in link step mode



Run-time replacing an LLI_n with a new LLI_n', in link step mode (in linked-list register file)

In this link step mode, during run-time, the software can build and insert a new LLI_n', after the HPDMA executed the transfer from the LLI_{n-1}, and loaded a formerly elaborated LLI_n from the memory by overwriting directly the linked-list register file with the new LLI_n', as shown in *Figure 92*.

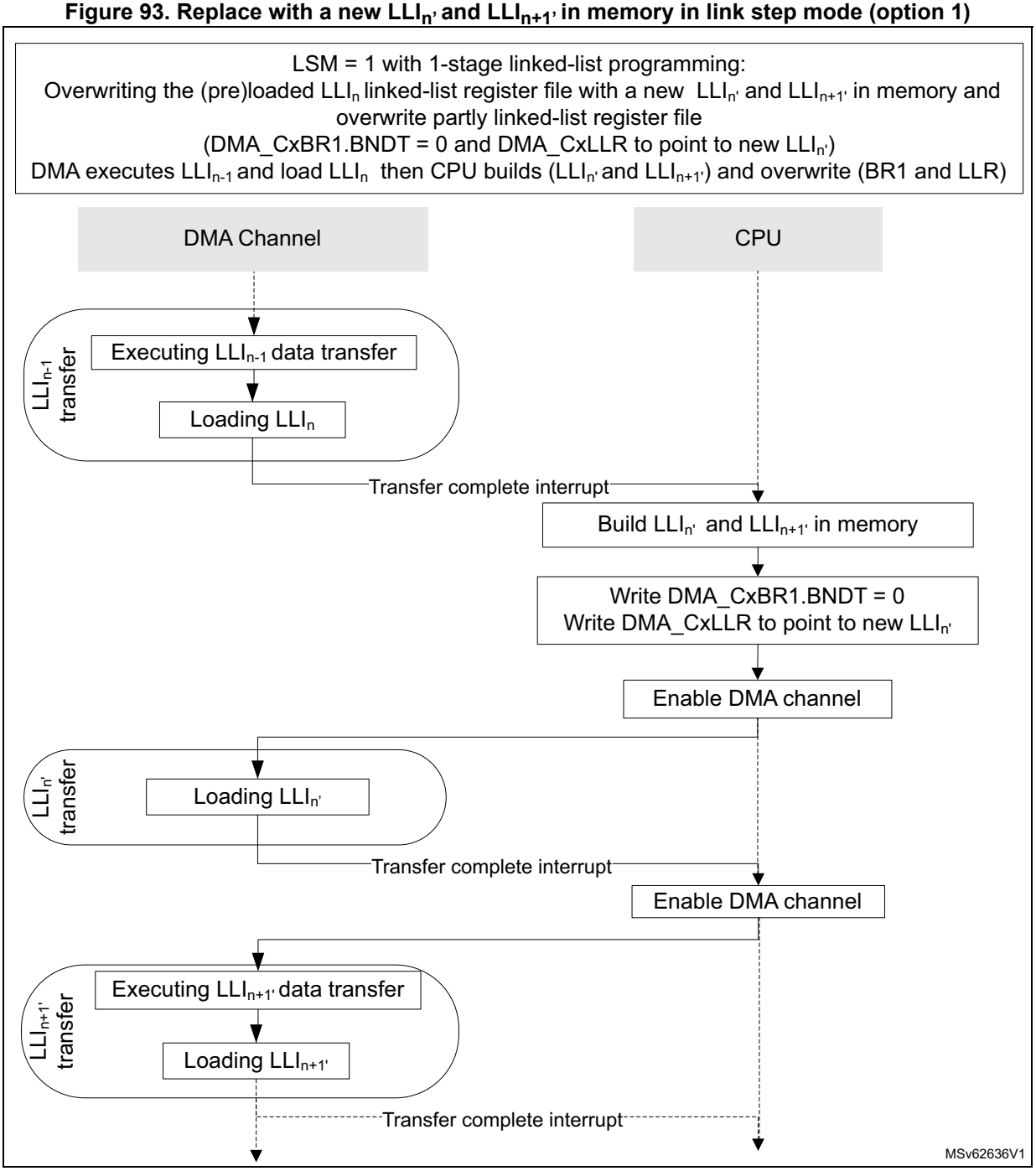
Figure 92. Replace with a new $LLI_{n'}$ in register file in link step mode



Run-time replacing an LLI_n with a new $LLI_{n'}$ in link step mode (in the memory)

The software can build and insert a new $LLI_{n'}$ and $LLI_{n+1'}$ in the memory, after HPDMA executed the transfer from the LLI_{n-1}, and loaded a formerly elaborated LLI_n from the memory, by overwriting partly the linked-list register file (HPDMA_CxBR1.BNDT[15:0] to be null, and HPDMA_CxLLR to point to new LLI_n) as shown in *Figure 93*.

Figure 93. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 1)



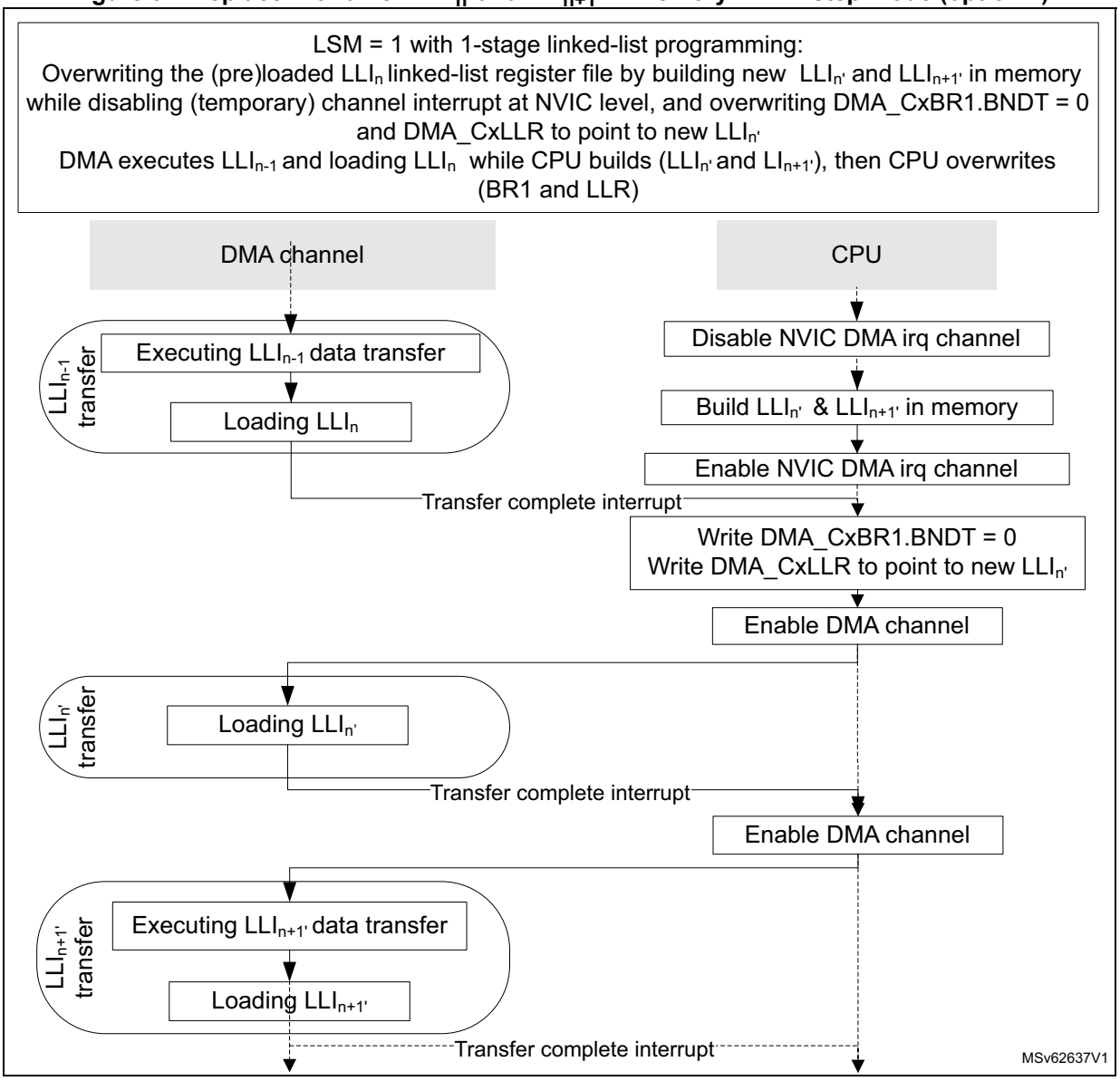
Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode

Other software implementations exist. Meanwhile the HPDMA executes the transfer from the LLI_{n-1} and loads a formerly elaborated LLI_n from the memory (or even earlier), the software can do the following:

1. Disable the NVIC for not being interrupted by the interrupt handling.
2. Build a new $LLI_{n'}$ and a new $LLI_{n+1'}$.
3. Enable again the NVIC for the channel interrupt (transfer complete) notification.

The software in the interrupt handler for LLI_{n-1} is then restricted to overwrite HPDMA_CxBR1.BNDT[15:0] to be null and HPDMA_CxLLR to point to new $LLI_{n'}$, as shown in the figure below.

Figure 94. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 2)



12.4.10 HPDMA channel state and linked-list programming

The software can reconfigure a channel when the channel is disabled (HPDMA_CxCR.EN = 0), and update the execution mode (HPDMA_CxCR.LSM) to change from/to run-to-completion mode to/from link step mode.

In any execution mode, the software can:

- Reprogram LLI_{n+1} in the memory to finally complete the channel by this LLI_{n+1} (clear the HPDMA_CxLLR of this LLI_{n+1}), before this LLI_{n+1} is loaded/used by the HPDMA channel
- Abort and reconfigure the channel with an LSM update (see *Section 12.4.4*.)

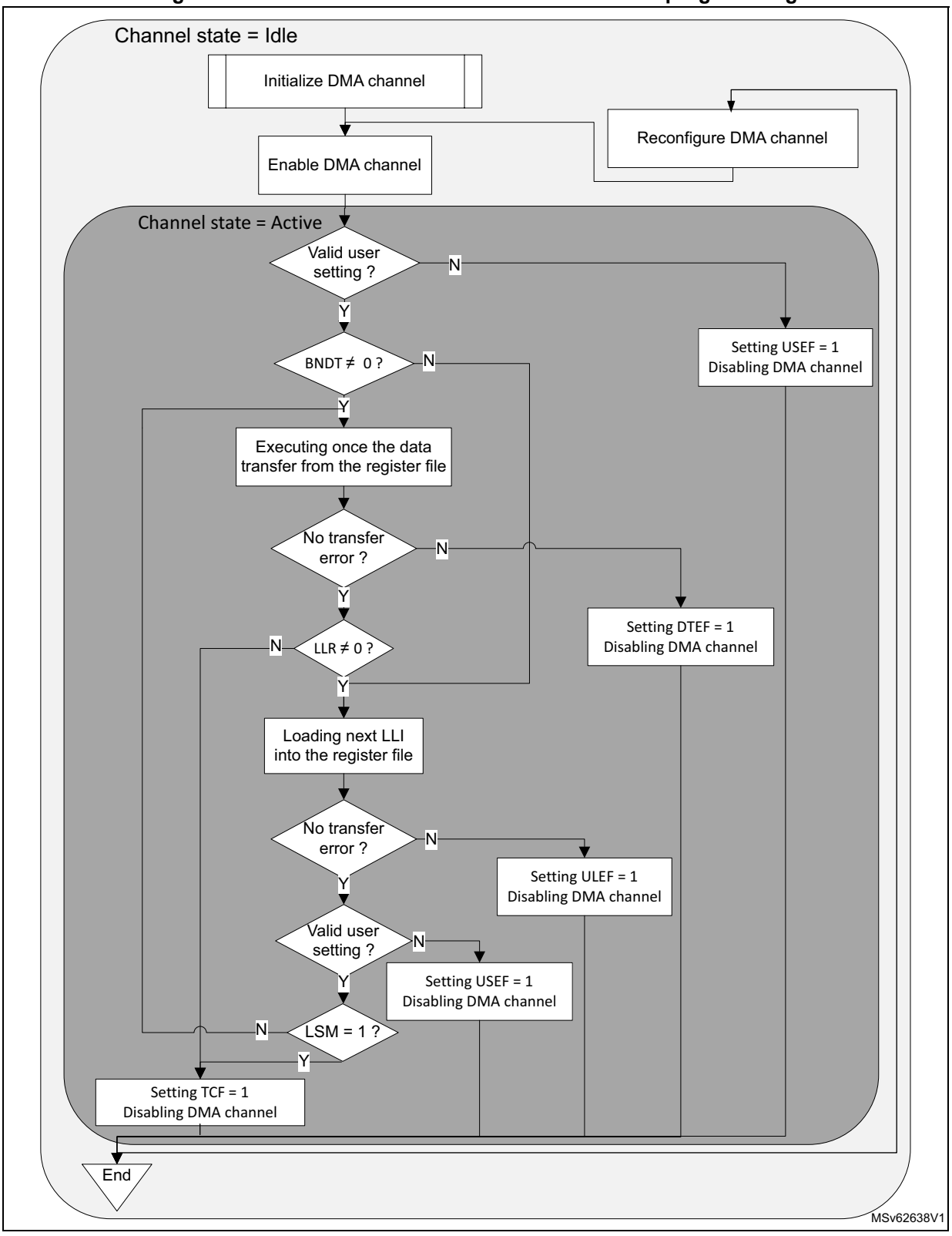
In link step mode, the software can clear LSM after each a single execution of any LLI, during LLI_{n-1}.

Figure 95 shows the overall and unified HPDMA linked-list programming, whatever is the execution mode.

Note: Figure 95 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in HPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion the hardware always set TCEF = 1 and disables the channel. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 95, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes BNDT ≠ 0 and PFREQ = 1 if the peripheral requests a data transfer with early termination (see Section 12.3.5).

Figure 95. HPDMA channel execution and linked-list programming



12.4.11 HPDMA FIFO-based transfers

There is a single transfer operation mode: the FIFO mode. There are FIFO-based transfers. Any channel x is implemented with a dedicated FIFO whose size is defined by dma_fifo_size[x] (see *Section 12.3.1* for more details).

HPDMA burst

A programmed transfer at the lowest level is an HPDMA burst.

An HPDMA burst is a burst of data received from the source, or a burst of data sent to the destination. A source (and destination) burst is programmed with a burst length by SBL_1[5:0] (respectively DBL_1[5:0]), and with a data width defined by SDW_LOG2[1:0] (respectively DDW_LOG2[1:0]) in HPDMA_CxTR1.

The addressing mode after each data (named beat) of an HPDMA burst is defined by SINC and DINC in HPDMA_CxTR1, for source and destination respectively: either a fixed addressing or an incremented addressing with contiguous data.

The start and next addresses of an HPDMA source/destination burst (defined by HPDMA_CxSAR and HPDMA_CxDAR) must be aligned with the respective data width.

The table below lists the main characteristics of an HPDMA burst.

Table 88. Programmed HPDMA source/destination burst

<table>
<tr><th>SAP/DAP (allocated port)</th><th>SDW_LOG2[1:0] DDW_LOG2[1:0]</th><th>Data width (bytes)</th><th>SINC/DINC</th><th>SBL_1[5:0] DBL_1[5:0]</th><th>Burst length (data/ beats)</th><th>Next data/ beat address</th><th>Next burst address</th><th>Burst address alignment</th></tr>
<tr><td rowspan="8">0: AXI
1: AHB</td><td>00</td><td>1</td><td rowspan="4">0 (fixed)</td><td rowspan="8">n = 0 to 63(1)</td><td rowspan="8">n+1</td><td rowspan="4">+ 0</td><td rowspan="4">+ 0</td><td>1</td></tr>
<tr><td>01</td><td>2</td><td>2</td></tr>
<tr><td>10</td><td>4</td><td>4</td></tr>
<tr><td>11(2)</td><td>8(2)</td><td>8(2)</td></tr>
<tr><td>00</td><td>1</td><td rowspan="4">1 (contiguously incremented)</td><td>+ 1</td><td>+ (n + 1)</td><td>1</td></tr>
<tr><td>01</td><td>2</td><td>+ 2</td><td>+ 2 * (n + 1)</td><td>2</td></tr>
<tr><td>10</td><td>4</td><td>+ 4</td><td>+ 4 * (n + 1)</td><td>4</td></tr>
<tr><td>11(2)</td><td>8(2)</td><td>+ 8(2)</td><td>+ 8 * (n + 1)(2)</td><td>8(2)</td></tr>
</table>

1. When S/DBL_1[5:0] = 0, burst is of length 1. Then burst can be also named as single.
2. Programming a source/destination data width of 8 bytes is allowed only if the allocated port (by SAP/DAP) is 64 bits. Otherwise, a user setting error is reported and no transfer is issued.

The next burst address in the above table is the next source/destination default address pointed by HPDMA_CxSAR or HPDMA_CxDAR, once the programmed source/destination burst is completed. This default value refers to the fixed/contiguously incremented address.

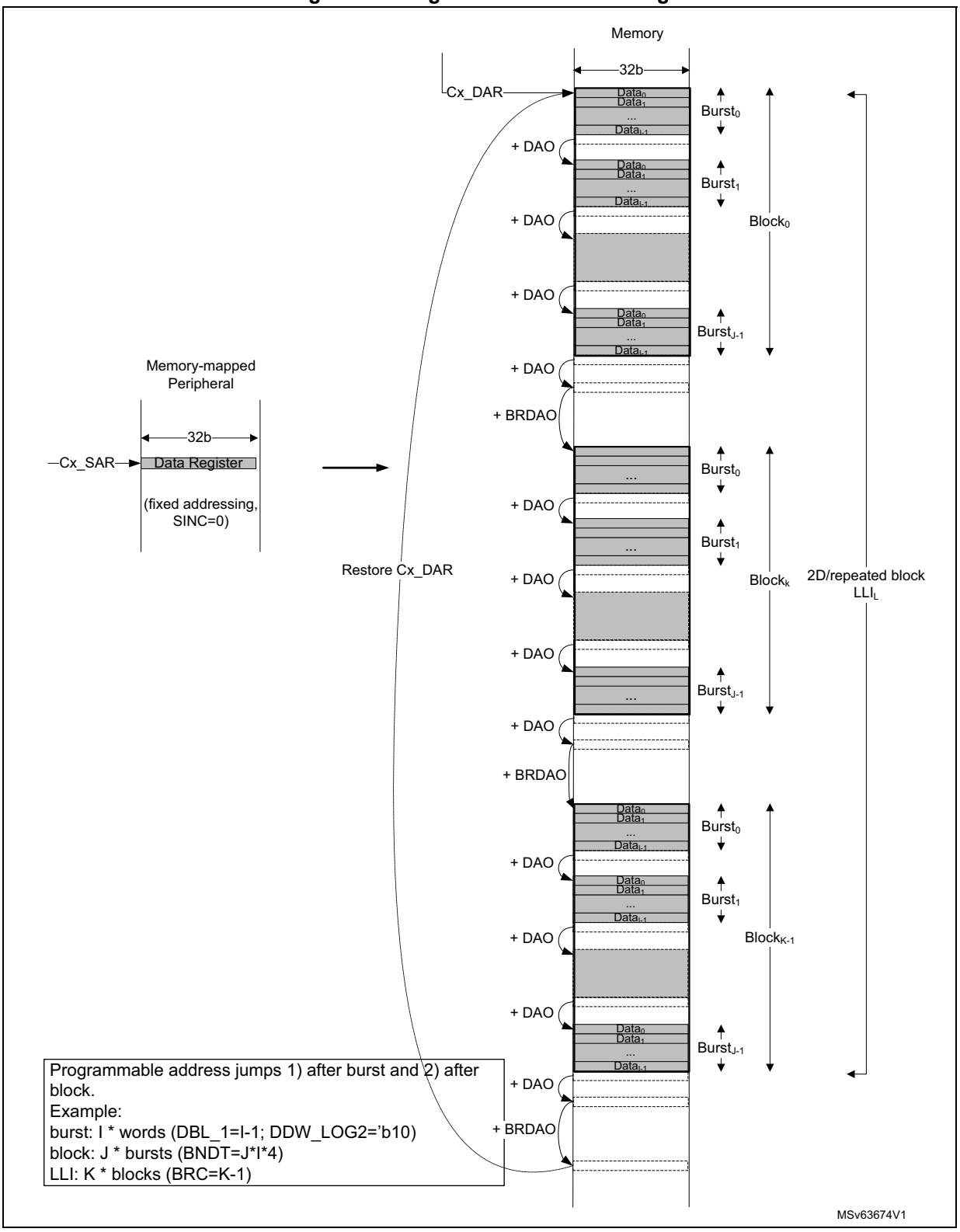
HPDMA burst with 2D addressing

When the channel has additional 2D addressing feature, this default value refers to the value without taking into account the two programmed incremented or decremented offsets. These two additional offsets (with a null default value) are applied:

- After each completed source/destination burst, as defined respectively by HPDMA_CxTR2.SAO[12:0]/DAO[12:0] and HPDMA_CxBR1.SDEC/DDEC
- After each completed block, as defined respectively by HPDMA_CxBR2.BRSAO[15:0]/BRDAO[15:0] and HPDMA_CxBR1.BRSDEC/BRDDEC)

Then, a 2D/repeated block can be addressed with a first programmed address jump after each completed burst, and with a second programmed address jump after each block, as depicted by *Figure 96* with a 2D destination buffer.

Figure 96. Programmed 2D addressing



HPDMA FIFO-based burst

In FIFO-mode, a transfer generally consists of two pipelined and separated burst transfers:

- One burst from the source to the FIFO over the allocated source master port, as defined by HPDMA_CxTR1.SAP
- One burst from the FIFO to the destination over the allocated destination master port, as defined by HPDMA_CxTR1.DAP

HPDMA source burst

The requested source burst transfer to the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to get one new burst from the source):

when FIFO level ≤ $2^{dma_fifo_size[x]}$ - (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- $2^{dma_fifo_size[x]}$ is half the FIFO size of channel x, in bytes (see *Section 12.3.1* for the implementation details and dma_fifo_size[x] value).
- (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$ is the size of the programmed source burst transfer, in bytes.

Based on the channel priority (HPDMA_CxCR.PRIO[1:0]), this ready FIFO-based source transfer is internally arbitrated versus the other requested and active channels.

HPDMA destination burst

The requested destination burst transfer from the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to push one new burst to the destination):

when FIFO level ≥ (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$ is the size of the programmed destination burst transfer, in bytes.

Based on the channel priority, this ready FIFO-based destination transfer is internally arbitrated versus the other requested and active channels.

HPDMA burst versus source block size, 1- or 4-Kbyte address boundary and FIFO size

The programmed source/destination HPDMA burst is implemented with an AHB/AXI burst as is, unless one of the following conditions is met:

- When half of the FIFO size of the channel x is lower than the programmed source/destination burst size, the programmed source/destination HPDMA burst is implemented with a series of singles or bursts of a lower size. Each transfer is of a size that is lower than or equal than half of the FIFO size, without any user constraint.
- If the source block size (HPDMA_CxBR1.BNDT[15:0]) is not a multiple of the source burst size but is a multiple of the data width of the source burst (HPDMA_CxTR1.SDW_LOG2[1:0]), the HPDMA modifies and shortens bursts into

singles or bursts of lower length, in order to transfer exactly the source block size, without any user constraint.

- If the source/destination burst transfer have crossed the 1- or 4-Kbyte address boundary on, respectively, an AHB or AXI transfer, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB/AXI protocol, without any user constraint.
- If the source/destination burst length exceeds 16 on a AHB transfer, or if the source/destination burst on an AXI transfer is both with fixed addressing and with a burst length which exceeds 16, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB/AXI protocol, without any user constraint.

In any case, the HPDMA keeps ensuring source/destination data (and address) integrity without any user constraint. The current FIFO level (software readable in HPDMA_CxSR) is compared to and updated with the effective transfer size, and the HPDMA re-arbitrates between each AHB/AXI single or burst transfer, possibly modified.

Based on the channel priority, each single or burst of a lower burst size versus the programmed burst, is internally arbitrated versus the other requested and active channels.

Note: In linked-list mode, the HPDMA read transfers related to the update of the linked-list parameters from the memory to the internal HPDMA registers, are scheduled over the link allocated port, as programmed by HPDMA_CxCR.LAP.

HPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension, and left/right alignment

The data handling is controlled by HPDMA_CxTR1. The source/destination data width of the programmed burst is byte, half-word, word, or double-word, as per SDW_LOG2[1:0] and DDW_LOG2[1:0] (see *Table 89*).

The user can configure the data handling between transferred data from the source and transfer to the destination. More specifically, programmed data handling is orderly performed with:

1. Byte-based source reordering
   - If SBX = 1 and if the source data width is a word or a double-word (for AXI source bus, SAP = 0), the two bytes of the unaligned half-word at the middle of each source data word are exchanged.
2. Data-width conversion by packing, unpacking, padding, or truncation, if destination data width is different than the source data width, depending on PAM[1:0]:
   - If destination data width > source data width, the post SBX source data is either right-aligned and padded with 0s, or sign extended up to the destination data width, or is FIFO queued and packed up to the destination data width.
   - If destination data width < source data width, the post SBX data is either right-aligned and left-truncated down to the destination data width, or is FIFO queued and unpacked and streamed down to the destination data width.

3. Byte-based destination re-ordering:
   - If DBX = 1 and if the destination data width is not a byte, the two bytes are exchanged within the aligned post PAM[1:0] half-words.
   - If DHX = 1 and if the destination data width is neither a byte nor a half-word, the two aligned half-words are exchanged within the aligned post PAM[1:0] words.
   - If DWX = 1 and if the destination data width is a double-word and if the selected destination port (via DAP) is 64-bit capable, the two aligned words are exchanged within aligned (post PAM[1:0]) double-words).

Note: Left-alignment with 0s-padding can be achieved by programming both a right-alignment with a 0s-padding, and a destination byte-based re-ordering.

The table below lists the possible data handling from the source to the destination.

Table 89. Programmed data handling

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
00	Byte	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$	x	00	Byte	xx	x	x	x	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, 0P)(3)(4)	0			$0B_3,0B_2,0B_1,0B_0$
							1			B_30,B_20,B_10,B_00
						01 (RA, SE)(3)(4)	0			SB_3,SB_2,SB_1,SB_0
							1			B_3S,B_2S,B_1S,B_0S
						1x (PACK)	0			$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1			$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)(3)(4)	0	0		$000B_1,000B_0$
							1			$00B_10,00B_00$
							0	1		$0B_100,0B_000$
							1			B_1000,B_0000
						01 (RA, SE)(3)(4)	0	0		$SSSB_1,SSSB_0$
							1			SSB_1S,SSB_0S
							0	1		SB_1SS,SB_0SS
							1			B_1SSS,B_0SSS
						1x (PACK)	0	0		$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7,B_0B_1B_2B_3$

Table 89. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
00	Byte	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$	x	11(5)	Double-word	00 (RA, 0P)(3)(4)	0	0	0	$0000000B_0$
							1			$000000B_00$
							0	1		$00000B_000$
							1			$0000B_0000$
							0	0	1	$000B_00000$
							1			$00B_000000$
							0	1		$0B_0000000$
							1			$B_00000000$
						01 (RA, SE)(3)(4)	0	0	0	$SSSSSSSB_0$
							1			$SSSSSSB_0S$
							0	1		$SSSSSB_0SS$
							1			$SSSSB_0SSS$
							0	0	1	$SSSB_0SSSS$
							1			SSB_0SSSSS
							0	1		$SB_0SSSSSS$
							1			$B_0SSSSSSS$
						1x (PACK)	0	0	0	$B_7B_6B_5B_4B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7B_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_0B_7B_6B_5B_4$
							1			$B_2B_3B_0B_1B_6B_7B_4B_5$
							0	1		$B_1B_0B_3B_2B_5B_4B_7B_6$
							1			$B_0B_1B_2B_3B_4B_5B_6B_7$

Table 89. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
01	Half-word	$B_7B_6,B_5B_4,$ B_3B_2,B_1B_0	x	00	Byte	00 (RA, LT)(3)	x	x	x	B_6,B_4,B_2,B_0
						01 (LA, RT)(3)				B_7,B_5,B_3,B_1
						1x (UNPACK)				$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	xx	0			$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1			$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)(3)(4)	0	0		$00B_3B_2,00B_1B_0$
							1			B_3B_200,B_1B_000
							0	1		B_2B_300,B_0B_100
							1			$00B_2B_3,00B_0B_1$
						01 (RA, SE)(3)(4)	0	0		SSB_3B_2,SSB_1B_0
							1			B_3B_2SS,B_1B_0SS
							0	1		B_2B_3SS,B_0B_1SS
							1			SSB_2B_3,SSB_0B_1
						1x (PACK)	0	0		$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7,B_0B_1B_2B_3$

Table 89. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
01	Half-word	B_7B_6,B_5B_4, B_3B_2,B_1B_0	x	11(5)	Double-word	00 (RA, 0P)(3)(4)	0	0	0	$000000B_1B_0$
							1			$000000B_0B_1$
							0	1		$0000B_1B_000$
							1			$0000B_1B_000$
							0	0	1	$00B_1B_00000$
							1			$00B_0B_10000$
							0	1		$B_1B_0000000$
							1			$B_0B_1000000$
						01 (RA, SE)(3)(4)	-	0	0	$SSSSSSB_1B_0$
							1			$SSSSSSB_0B_1$
							0	1		$SSSSB_1B_0SS$
							1			$SSSSB_1B_0SS$
							0	0	1	SSB_1B_0SSSS
							1			SSB_0B_1SSSS
							0	1		$B_1B_0SSSSSS$
							1			$B_0B_1SSSSSS$
						1x (PACK)	0	0	0	$B_7B_6B_5B_4B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7B_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_0B_7B_6B_5B_4$
							1			$B_2B_3B_0B_1B_6B_7B_4B_5$
							0	1		$B_1B_0B_3B_2B_5B_4B_7B_6$
							1			$B_0B_1B_2B_3B_4B_5B_6B_7$

Table 89. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	0	00	Byte	00 (RA, LT)(3)	-	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)(3)	-			B_{15},B_{11},B_7,B_3
						1x (UNPACK)	x			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, LT)(3)	0			B_5B_4,B_1B_0
							1			B_4B_5,B_0B_1
						01 (LA, RT)(3)	0			B_7B_6,B_3B_2
							1			B_6B_7,B_2B_3
						1x (UNPACK)	0			$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1			$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	xx	0	0		$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7,B_0B_1B_2B_3$

Table 89. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	0	11(5)	Double-word	1x (PACK)	0	0	0	$B_7B_6B_5B_4B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7B_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_0B_7B_6B_5B_4$
							1			$B_2B_3B_0B_1B_6B_7B_4B_5$
							0	1		$B_1B_0B_3B_2B_5B_4B_7B_6$
							1			$B_0B_1B_2B_3B_4B_5B_6B_7$
						00 (RA, 0P)(3)(4)	0	0	0	$0000B_3B_2B_1B_0$
							1			$0000B_2B_3B_0B_1$
							0	1		$0000B_1B_0B_3B_2$
							1			$0000B_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_00000$
							1			$B_1B_0B_3B_20000$
							0	1		$B_3B_2B_1B_00000$
							1			$B_0B_1B_2B_30000$
						01 (RA, SE)(3)(4)	0	0	0	$SSSSB_3B_2B_1B_0$
							1			$SSSSB_2B_3B_0B_1$
							0	1		$SSSSB_1B_0B_3B_2$
							1			$SSSSB_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_0SSSS$
							1			$B_1B_0B_3B_2SSSS$
							0	1		$B_3B_2B_1B_0SSSS$
							1			$B_0B_1B_2B_3SSSS$

Table 89. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	1	00	Byte	00 (RA, LT)(3)	-	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)(3)	-			B_{15},B_{11},B_7,B_3
						1x (UNPACK)	x			$B_7,B_5,B_6,B_4,B_3,B_1,B_2,B_0$
				01	Half-word	00 (RA, LT)(3)	0			B_6B_4,B_2B_0
							1			B_4B_6,B_0B_2
						01 (LA, RT)(3)	0			B_7B_5,B_3B_1
							1			B_5B_7,B_1B_3
						1x (UNPACK)	0			$B_7B_5,B_6B_4,B_3B_1,B_2B_0$
							1			$B_5B_7,B_4B_6,B_1B_3,B_0B_2$
				10	Word	xx	0	0		$B_7B_5B_6B_4,B_3B_1B_2B_0$
							1			$B_5B_7B_4B_6,B_1B_3B_0B_2$
							0	1		$B_6B_4B_7B_5,B_2B_0B_3B_1$
							1			$B_4B_6B_5B_7,B_0B_2B_1B_3$

Table 89. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	1	11(5)	Double-word	00 (RA, 0P)(3)(4)	0	0	0	$0000B_3B_1B_2B_0$
							1			$0000B_1B_3B_0B_2$
							0	1		$0000B_2B_0B_3B_1$
							1			$0000B_0B_2B_1B_3$
							0	0	1	$B_3B_1B_2B_00000$
							1			$B_1B_3B_0B_20000$
							0	1		$B_2B_0B_3B_10000$
							1			$B_0B_2B_1B_30000$
						01 (RA, SE)(3)(4)	0	0	0	$SSSSB_3B_1B_2B_0$
							1			$SSSSB_1B_3B_0B_2$
							0	1		$SSSSB_2B_0B_3B_1$
							1			$SSSSB_0B_2B_1B_3$
							0	0	1	$B_3B_1B_2B_0SSSS$
							1			$B_1B_3B_0B_2SSSS$
							0	1		$B_2B_0B_3B_1SSSS$
							1			$B_0B_2B_1B_3SSSS$
						1x (PACK)	0	0	0	$B_7B_5B_6B_4B_3B_1B_2B_0$
							1			$B_5B_7B_4B_6B_1B_3B_0B_2$
							0	1		$B_6B_4B_7B_5B_2B_0B_3B_1$
							1			$B_4B_6B_5B_7B_0B_2B_1B_3$
							0	0	1	$B_3B_1B_2B_0B_7B_5B_6B_4$
							1			$B_1B_3B_0B_2B_5B_7B_4B_6$
							0	1		$B_2B_0B_3B_1B_6B_4B_7B_5$
							1			$B_0B_2B_1B_3B_4B_6B_5B_7$

Table 89. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
11(6)	Double-word	$B_7B_6B_5B_4B_3B_2B_1B_0$	0	00	Byte	00 (RA, LT)(3)	-	x	x	B_{24},B_{16},B_8,B_0
						01 (LA, RT)(3)	-			B_{31},B_{23},B_{15},B_7
						1x (UNPACK)	x			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, LT)(3)	0			B_9B_8,B_1B_0
							1			B_8B_9,B_0B_1
						01 (LA, RT)(3)	0			$B_{15}B_{14},B_7B_6$
							1			$B_{14}B_{15},B_6B_7$
						1x (UNPACK)	0			$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1			$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, LT)(3)	0	0		$B_{11}B_{10}B_9B_8,$ $B_3B_2B_1B_0$
							1			$B_{10}B_{11}B_8B_9,$ $B_2B_3B_0B_1$
							0	1		$B_9B_8B_{11}B_{10},$ $B_1B_0B_3B_2$
							1			$B_8B_9B_{10}B_{11},$ $B_0B_1B_2B_3$
						01 (LA, RT)(3)	0	0		$B_{15}B_{14}B_{13}B_{12},$ $B_7B_6B_5B_4$
							1			$B_{14}B_{15}B_{12}B_{13},$ $B_6B_7B_4B_5$
							0	1		$B_{13}B_{12}B_{15}B_{14},$ $B_5B_4B_7B_6$
							1			$B_{12}B_{13}B_{14}B_{15},$ $B_4B_5B_6B_7$
						1x (UNPACK)	0	0		$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7,B_0B_1B_2B_3$

Table 89. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
11(6)	Double-word	$B_7B_6B_5B_4B_3B_2B_1B_0$	0	11(5)	Double-word	xx	0	0	0	$B_7B_6B_5B_4B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7B_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_0B_7B_6B_5B_4$
							1			$B_2B_3B_0B_1B_6B_7B_4B_5$
							0	1		$B_1B_0B_3B_2B_5B_4B_7B_6$
							1			$B_0B_1B_2B_3B_4B_5B_6B_7$
			1	00	Byte	00 (RA, LT)(3)	x	x	x	B_{24},B_{16},B_8,B_0
						01 (LA, RT)(3)				B_{31},B_{23},B_{15},B_7
						1x (UNPACK)				$B_7,B_5,B_6,B_4,B_3,B_1,B_2,B_0$
				01	Half-word	00 (RA, LT)(3)	0			$B_{10}B_8,B_2B_0$
							1			B_8B_{10},B_0B_2
						01 (LA, RT)(3)	0			$B_{15}B_{13},B_7B_5$
							1			$B_{13}B_{15},B_5B_7$
						1x (UNPACK)	0			$B_7B_5,B_6B_4,B_3B_1,B_2B_0$
							1			$B_5B_7,B_4B_6,B_1B_3,B_0B_2$

Table 89. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
11(6)	Double-word	$B_7B_6B_5B_4B_3B_2B_1B_0$	1	10	Word	00 (RA, LT)(3)	0	0	x	$B_{11}B_9B_{10}B_8,B_3B_1B_2B_0$
							1			$B_9B_{11}B_8B_{10},B_1B_3B_0B_2$
							0	1		$B_{10}B_8B_{11}B_9,B_2B_0B_3B_1$
							1			$B_8B_{10}B_9B_{11},B_0B_2B_1B_3$
						01 (LA, RT)(3)	0	0		$B_{15}B_{13}B_{14}B_{12}$, $B_7B_5B_6B_4$
							1			$B_{13}B_{15}B_{12}B_{14}$, $B_5B_7B_4B_6$
							0	1		$B_{14}B_{12}B_{15}B_{13}$, $B_6B_4B_7B_5$
							1			$B_{12}B_{14}B_{13}B_{15}$, $B_4B_6B_5B_7$
						1x (UNPACK)	0	0		$B_7B_5B_6B_4,B_3B_1B_2B_0$
							1			$B_5B_7B_4B_6,B_1B_3B_0B_2$
							0	1		$B_6B_4B_7B_5,B_2B_0B_3B_1$
							1			$B_4B_6B_5B_7,B_0B_2B_1B_3$
				11(5)	Double-word	xx	0	0	0	$B_7B_5B_6B_4B_3B_1B_2B_0$
							1			$B_5B_7B_4B_6B_1B_3B_0B_2$
							0	1		$B_6B_4B_7B_5B_2B_0B_3B_1$
							1			$B_4B_6B_5B_7B_0B_2B_1B_3$
							0	0	1	$B_3B_1B_2B_0B_7B_5B_6B_4$
							1			$B_1B_3B_0B_2B_5B_7B_4B_6$
							0	1		$B_2B_0B_3B_1B_6B_4B_7B_5$
							1			$B_0B_2B_1B_3B_4B_6B_5B_7$

1. Data stream is timely ordered starting from the byte with the lowest index (B_0).
2. RA= right aligned, LA = left aligned, RT = right truncated, LT = left truncated, 0P = zero bit padding up to the destination data width, SE = sign bit extended up to the destination data width.
3. RA= right aligned. LA = left aligned. RT = right truncated. LT = left truncated.
4. 0P= zero-bit padding up to the destination data width. SE = sign bit extended up to the destination data width.
5. if DDW_LOG2[1:0] = 11 and if the selected destination port (via DAP) is 64-bit capable. If DDW_LOG2[1:0] = 11 and the selected destination port is not 64-bit capable, a user setting error (USEF) is reported.
6. if SDW_LOG2[1:0] = 11 and if the selected source port (via SAP) is 64-bit capable. If SDW_LOG2[1:0] = 11 and the selected source port is not 64-bit capable, a user setting error (USEF) is reported.

12.4.12 HPDMA transfer request and arbitration

HPDMA transfer request

As defined by HPDMA_CxTR2, a programmed HPDMA data transfer is requested with one of the following:

- a software request if the control bit SWREQ = 1: This is used typically by the CPU for a data transfer from a memory-mapped address to another memory mapped address (memory-to-memory, GPIO to/from memory)
- an input hardware request coming from a peripheral if SWREQ = 0: The selection of the HPDMA hardware peripheral request is driven by REQSEL[7:0] (see *Section 12.3.4*). The selected hardware request can be one of the following:
  - An hardware request from a peripheral configured in HPDMA mode (for a transfer from/to the peripheral data register respectively to/from the memory)
  - An hardware request from a peripheral for its control register update from the memory
  - An hardware request from a peripheral for a read of its status registers transferred to the memory

Caution: The user must not assign the same input hardware peripheral HPDMA request via HPDMA_CxTR.REQSEL[7:0] to two different channels, if at a given time this request is asserted by the peripheral, and each channel is ready to execute this requested data transfer. There is no user setting error reporting.

HPDMA transfer request for arbitration

A ready FIFO-based HPDMA source single/burst transfer (from the source address to the FIFO) to be scheduled over the allocated master port (HPDMA_CxTR1.SAP) is arbitrated based on the channel priority (HPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested HPDMA transfers to the same master port.

A ready FIFO-based HPDMA destination single/burst transfer (from the FIFO to the destination address) to be scheduled over the allocated master port (HPDMA_CxTR1.DAP) is arbitrated based on the channel priority (HPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested HPDMA transfers to the same master port.

An arbitrated HPDMA requested link transfer consists of one 32-bit read from the linked-list data structure in the memory to one of the linked-list registers (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, HPDMA_CxTR3, HPDMA_CxBR2 or HPDMA_CxLLR). Each 32-bit read from the memory is arbitrated with the same channel priority as for data transfers, in order to be scheduled over the allocated master port (HPDMA_CxCR.LAP).

Whatever the requested data transfer is programmed with a software request for a memory-to- memory transfer (HPDMA_CxTR2.SWREQ = 1), or with a hardware request (HPDMA_CxTR2.SWREQ = 0) for a memory-to-peripheral transfer or a peripheral-to-memory transfer and whatever is the hardware request type, re-arbitration occurs after each granted single/burst transfer.

When an hardware request is programmed from a destination peripheral (HPDMA_CxTR2.SWREQ = 0 and HPDMA_CxTR2.DREQ = 1), the first memory read of a (possibly 2D/repeated) block (the first ready FIFO-based source burst request), is gated by the occurrence of the corresponding and selected hardware request. This first read request

to memory is not taken into account earlier by the arbiter (not as soon as the block transfer is enabled and executable).

HPDMA arbitration

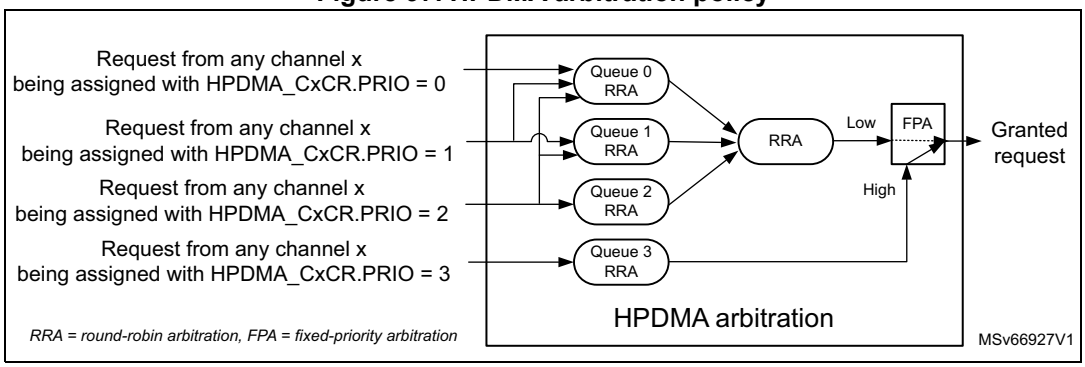
The HPDMA arbitration is directed from the 4-grade assigned channel priority (HPDMA_CxCR.PRIO[1:0]). The arbitration policy, as illustrated in *Figure 97*, is defined by:

- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels

  This traffic class is granted via a fixed-priority arbitration against any other low-priority traffic class. Within this class, requested single/burst transfers are round-robin arbitrated.
- three low-priority traffic classes (queues 0, 1 or 2) for non time-sensitive channels with priority 0, 1 or 2

  Each requested single/burst transfer within this class is round-robin arbitrated, with a weight that is monotonically driven from the programmed priority:
  - Requests with priority 0 are allocated to the queue 0.
  - Requests with priority 1 are allocated and replicated to the queue 0 and queue 1.
  - Requests with priority 2 are allocated and replicated to the queue 0, queue 1, and queue 2.
  - Any queue 0, 1 or 2 equally grants any of its active input requests in a round-robin manner, provided there are simultaneous requests.
  - Additionally, there is a second stage for the low-traffic with a round-robin arbiter that fairly alternates between simultaneous selected requests from queue 0, queue 1 and queue 2.

Figure 97. HPDMA arbitration policy



HPDMA arbitration and bandwidth

With this arbitration policy, the following is guaranteed:

- Equal maximum bandwidth between requests with same priority
- Reserved bandwidth (noted as B_{Q3}) to the time-sensitive requests (with priority 3)
- Residual weighted bandwidth between different low-priority requests (priority 0 versus priority 1 versus priority 2).

The two following examples highlight that the weighted round-robin arbitration is driven by the programmed priorities:

- **Example 1**: basic application with two non time-sensitive HPDMA requests: req0 and req1. There are the following programming possibilities:
  - If they are assigned with same priority, the allocated bandwidth by the arbiter to req0 (B_{req0}) is **equal** to the allocated bandwidth to req1(B_{req1}).

    $B_{req0} = B_{req1} = 1/2 * (1 - B_{Q3})$
  - If req0 is assigned to priority 0 and req1 to priority 1, the allocated bandwidth to req0 (B_{P0}) is **3 times less** than the allocated bandwidth to req1 (B_{P1}).

    $B_{req0} = B_{P0} = 1/2 * 1/2 * (1 - B_{Q3}) = 1/4 * (1 - B_{Q3})$

    $B_{req1} = B_{P1} = (1/2 + 1) * 1/2 * (1 - B_{Q3}) = 3/4 * (1 - B_{Q3})$
  - If req0 is assigned to priority 0 and req1 to priority 2, the allocated bandwidth to req0 (B_{P0}) is **5 times less** than the allocated bandwidth to req1 (B_{P2}).

    $B_{req0} = B_{P0} = 1/2 * 1/3 * (1 - B_{Q3}) = 1/6 * (1 - B_{Q3})$

    $B_{req1} = B_{P2} = (1/2 + 1 + 1) * 1/3 * (1 - B_{Q3}) = 5/6 * (1 - B_{Q3})$

  The above computed bandwidth calculation is based on a theoretical input request, always active for any HPDMA clock cycle. This computed bandwidth from the arbiter must be weighted by the frequency of the request given by the application, that cannot be always active and may be quite much variable from one HPDMA client (example I2C at 400 kHz) to another one (PWM at 1 kHz) than the above x3 and x5 ratios.
- **Example 2**: application where the user distributes a same non-null N number of HPDMA requests to every non time-sensitive priority 0, 1 and 2. The bandwidth calculation is then the following:
  - The allocated bandwidth to the set of requests of priority 0 (B_{P0}) is

    $B_{P0} = 1/3 * 1/3 * (1 - B_{Q3}) = 1/9 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 1 (B_{P1}) is

    $B_{P1} = (1/3 + 1/2) * 1/3 * (1 - B_{Q3}) = 5/18 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 2 (B_{P2}) is

    $B_{P2} = (1/3 + 1/2 + 1) * 1/3 * (1 - B_{Q3}) = 11/18 * (1 - B_{Q3})$
  - The allocated bandwidth to any request n ($B_{n)}$ among the N requests of that priority Pi (i = 0 to 2) is $B_n = 1/N * B_{Pi}$
  - The allocated bandwidth to any request n of priority i ($B_{n, Pi}$) is

    $B_{n, P0} = 1/N * 1/9 * (1 - B_{Q3})$

    $B_{n, P1} = 1/N * 5/18 * (1 - B_{Q3})$

    $B_{n, P2} = 1/N * 11/18 * (1 - B_{Q3})$

  In this example, when the master port bus bandwidth is not totally consumed by the time-sensitive queue 3, the residual bandwidth is such that 2.5 times less bandwidth is allocated to any request of priority 0 versus priority 1, and 5.5 times less bandwidth is allocated to any request of priority 0 versus priority 2.

More generally, assume that the following requests are present:

- I requests (I ≥ 0) assigned to priority 0
  If I > 0, these requests are noted from i = 0 to I-1.
- J requests (J ≥ 0) assigned to priority 1
  If J > 0, these requests are noted from j = 0 to J-1.
- K requests (K > 0) assigned to priority 2
  These requests are noted from k = 0 to K-1
- L requests (L ≥ 0) assigned to priority 3
  If L > 0, these requests are noted from l = 0 to L-1.

As B_{Q3} is the reserved bandwidth to time-sensitive requests, the bandwidth for each request L with priority 3 is:

- $B_l = B_{Q3} / L$ for L > 0 (else: $B_l = 0$)

The bandwidth for each non-time sensitive queue is:

- $B_{Q0} = 1/3 * (1 - B_{Q3})$
- $B_{Q1} = 1/3 * (1 - B_{Q3})$
- $B_{Q2} = 1/3 * (1 - B_{Q3})$

The bandwidth for the set of requests with priority 0 is:

- $B_{P0} = I / (I + J + K) * B_{Q0}$

The bandwidth for each request i with priority 0 is:

- $B_i = B_{P0} / I$ for L > 0 (else $B_{P0} = 0$)

The bandwidth for the set of requests with priority 1 and routed to queue 0 is:

- $B_{P1,Q0} = J / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 1 and routed to queue 1 is:

- $B_{P1,Q1} = J / (J + K) * B_{Q1}$

The total bandwidth for the set of requests with priority 1 is:

- $B_{P1} = B_{P1,Q0} + B_{P1,Q1}$

The bandwidth for each request j with priority 1 is:

- $B_j = B_{P1} / J$ for J > 0 (else $B_j = 0$)

The bandwidth for the set of requests with priority 2 and routed to queue 0 is:

- $B_{P2,Q0} = K / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 2 and routed to queue 1 is:

- $B_{P2,Q1} = K / (J + K) * B_{Q1}$

The bandwidth for the set of requests with priority 2 and routed to queue 2 is:

- $B_{P2,Q2} = B_{Q2}$

The total bandwidth for the set of requests with priority 2 is:

- $B_{P2} = B_{P2,Q0} + B_{P2,Q1} + B_{P2,Q2}$

The bandwidth for each request k with priority 2 is:

- $B_k = B_{P2} / K$ (K>0 in the general case)

Thus finally the maximum allocated residual bandwidths for any i, j, k non-time sensitive request are:

- in the general case (when there is at least one request k with a priority 2 (K > 0)):
  - $B_i = 1/I * 1/3 * I/(I + J + K) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/3 *[J/(I + J + K) + J/(J + K)] * (1 - B_{Q3})$
  - $B_k = 1/K * 1/3 *[K/(I + J + K) + K/(J + K) + 1] * (1 - B_{Q3})$
- in the specific case (when there is no request k with a priority 2 (K = 0)):
  - $B_i = 1/I * 1/2 * I/(I + J) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/2 *[J/(I + J) + 1] * (1 - B_{Q3})$

Consequently, the HPDMA arbiter can be used as a programmable weighted bandwidth limiter, for each queue and more generally for each request/channel. The different weights are monotonically resulting from the programmed channel priorities.

12.4.13 HPDMA triggered transfer

A programmed HPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event, as defined by HPDMA_CxTR2.TRIGPOL[1:0] and HPDMA_CxTR2.TRIGSEL[5:0] (see *Section 12.3.6* for the trigger selection).

The triggered transfer, as defined by the trigger mode in HPDMA_CxTR2.TRIGM[1:0], can be at LLI data transfer level, to condition the first burst read of a block, the first burst read of a 2D/repeated block (for channel x = 12 to 15), or each programmed burst read/write. The trigger mode can also be programmed to condition the LLI link transfer (see TRIGM[1:0] in *HPDMA channel x transfer register 2 (HPDMA_CxTR2)* for more details).

Trigger hit memorization and trigger overrun flag generation

The HPDMA monitoring of a trigger for a channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (respectively TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

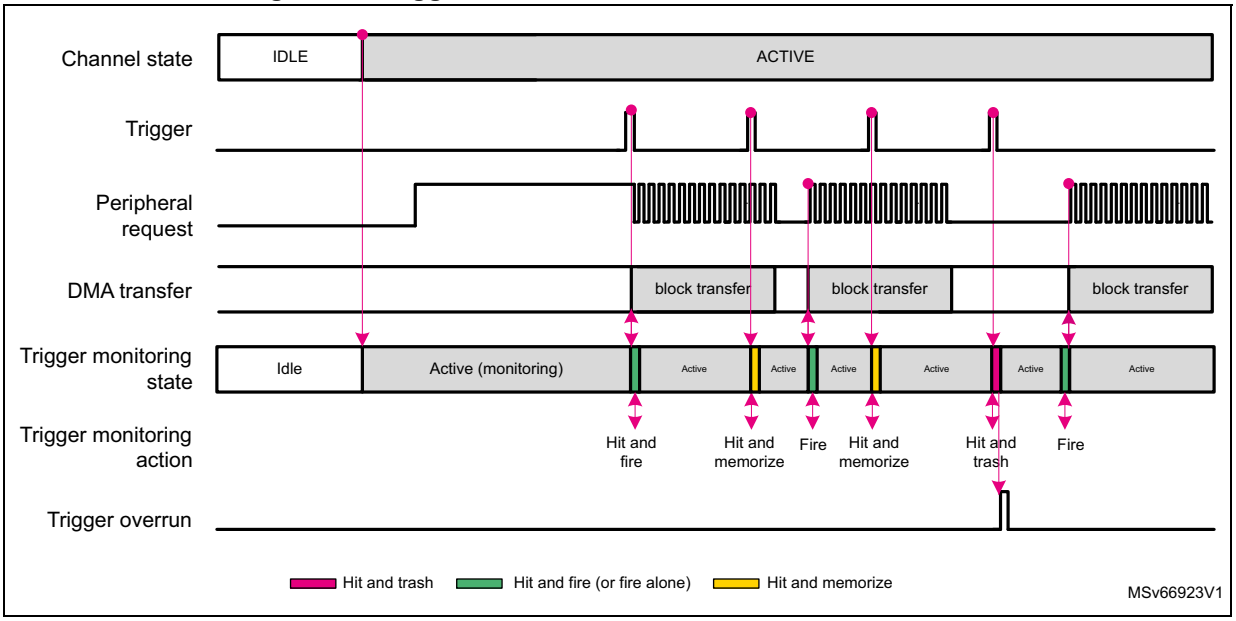
The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[5:0] are not modified, and the channel is enabled.

Transferring a next $LLI_{n+1,}$ that updates HPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the possible memorized hit of the formerly defined LLI_n trigger.

Caution: After a first new trigger, hit_{n+1} is memorized. If another trigger hit_{n+2} is detected, and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (HPDMA_CxSR.TOF = 1) and an interrupt is generated if enabled (if HPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

The figure below illustrates the trigger hit, memorization, and overrun in the configuration example with a block-level trigger mode and a rising edge trigger polarity.

Figure 98. Trigger hit, memorization and overrun waveform



Note: The user can assign the same input trigger event to different channels. This can be used to trigger different channels on a broadcast trigger event.

12.4.14 HPDMA circular buffering with linked-list programming

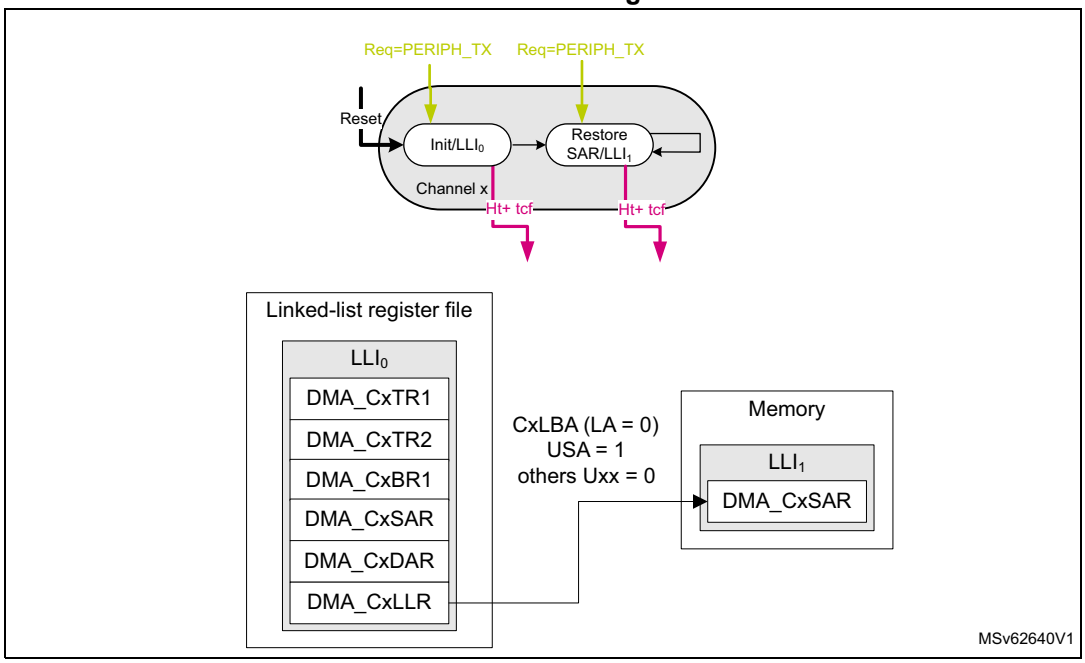
HPDMA circular buffering for memory-to-peripheral and peripheral-to-memory transfers, with a linear addressing channel

For a circular buffering, with a continuous memory-to-peripheral (or peripheral-to-memory) transfer, the software must set up a channel with half-transfer and complete-transfer event/interrupt generation (HPDMA_CxCR.HTIE = 1 and HPDMA_CxCR.TCIE = 1), in order to enable a concurrent buffer software processing.

LLI_0 is configured for the first block transfer with the linear addressing channel.
A continuously-executed LLI_1 is needed to restore the memory source (or destination) start address for the memory-to-peripheral transfer (respectively the peripheral-to-memory).
The HPDMA automatically reloads the initially programmed HPDMA_CxBR1.BNDT[15:0] when a block transfer is completed, and there is no need to restore HPDMA_CxBR1.

The figure below illustrates this programming with a linear addressing HPDMA channel and a source circular buffer.

Figure 99. HPDMA circular buffer programming: update of the memory start address with a linear addressing channel

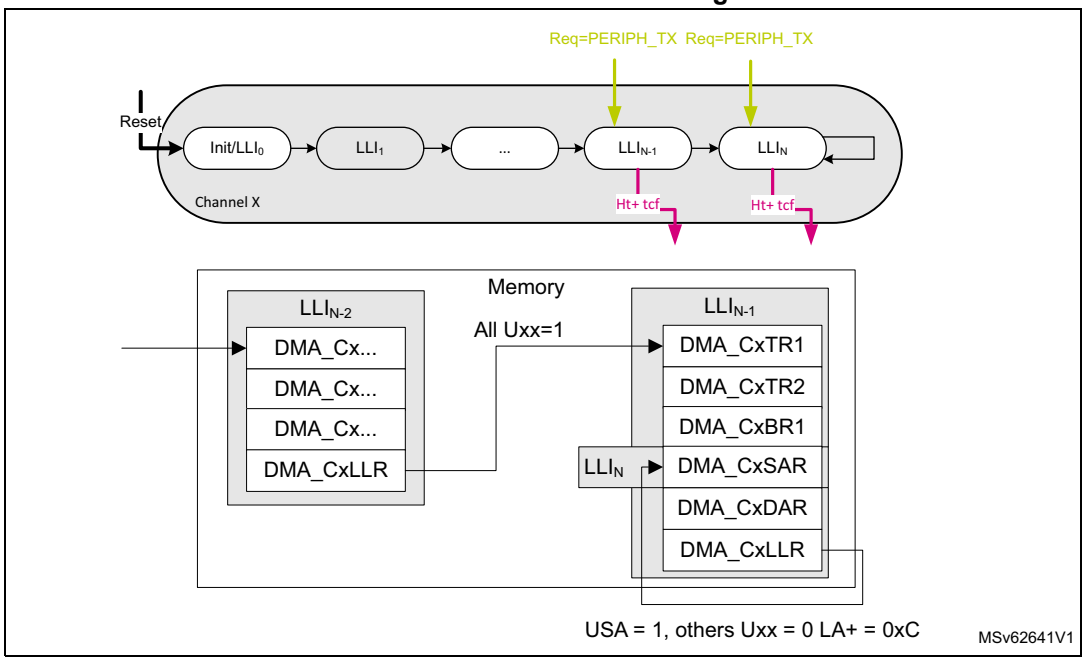


Note: With a 2D addressing channel, a single LLI can be used with HPDMA_CxBR1.BRC[10:0] = 1. The user can program a negative memory block address offset with HDMA_CxBR2 and HDMA_CxBR1, in order to jump back to the memory source or destination start address.

If the circular buffering must be executed after some other transfers over the shared HPDMA channel x, the before-last LLI_{N-1} in the memory is needed to configure the first block transfer. The last LLI_N restores the memory source (or destination) start address in memory-to-peripheral transfer (respectively in peripheral-to-memory).

The figure below illustrates this programming with a linear addressing shared HPDMA channel, and a source circular buffer.

Figure 100. Shared HPDMA channel with circular buffering: update of the memory start address with a linear addressing channel



12.4.15 HPDMA transfer in peripheral flow-control mode

A peripheral with the peripheral flow-control mode feature can decide to early terminate an HPDMA block transfer, provided that the allocated channel is implemented with this feature (see *Section 12.3.5*).

If the related HPDMA channel x is also programmed in peripheral flow-control mode (HPDMA_CxTR2.PFREQ = 1):

- The HPDMA block transfer starts as follows:
  - If HPDMA_CxBR1.BNDT[15:0] ≠ 0, the programmed value is internally taken into account by the HPDMA hardware.
  - If HPDMA_CxBR1.BNDT[15:0] = 0, the HPDMA hardware internally considers a 64-Kbyte value for the maximum source block size to be transferred.
- The HPDMA block transfer is completed as soon as the first occurrence of any of the following condition occurs:
  - When HPDMA_CxBR1.BNDT[15:0] = 0
  - When the peripheral early terminates the block. The complete transfer event is generated if programmed, depending on HPDMA_CxTR2. Then the software can read the current number of transferred bytes from the source (HPDMA_CxBR1.BNDT[15:0]), and/or read the current source or destination address of the buffer in memory (HPDMA_CxSAR[31:0] or HPDMA_CxDAR[31:0]).

In peripheral flow-control mode:

- A destination peripheral with an hardware requested transfer is not supported: memory-to-peripheral transfer is not supported.
- Data packing from a source peripheral is not supported.
- 2D/repeated block is not supported.
- HPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

12.4.16 HPDMA byte-based padding at the block level

For a given channel x, a LLI_n data transfer consists of a block transfer and is defined by the source block size HPDMA_CxBR1.BNDT[15:0], the size of the block of data to be transfered from the source, in bytes.

During a peripheral-to-memory transfer, a peripheral may be streaming and receiving a given number of bytes (for example, via an I2C, UART/USART/LPUART, ADC, DAC) or half-words, and this block size may not be a multiple of a word or a double-word. It is important to allow the user to program any number of bytes for the HPDMA source block size independently from the source data width and the destination data width, in order to use efficiently the bandwidth via the data width of the AHB system bus and/or the AXI system bus when accessing and writing into memory (embedded SRAM, NVM or external memory). This enables to decrease the bus bandwidth for the memory transfers by a ratio of up to 4 in 32-bit AHB and up to 8 in 64-bit AXI system. More generally, having the source block size independent from the source data width and the destination data width can be useful also with memory-to-peripheral transfers, and with memory-to-memory transfers (AHB from/to AXI).

A FIFO-based channel is able to perform byte-based padding on the last write(s) up to the fully transferred source block size. Then channel FIFO is empty. Data handling is performed on zero bytes in the same way as on regular data bytes from the source, as explained in *HPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension, and left/right alignment*.

There is byte-based padding at the block level:

1. If source data width is less than the destination data width, and if FIFO queuing and data packing is configured (PAM[1:0] = 1x):

- Padding with zero bytes. For example, if BNDT= 5, 8-bit reads, 32-bit writes:
  - If DBX = 0, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> 000B_4, $B_3B_2B_1B_0$
  - If DBX = 0, DHX = 1: B_4,B_3,B_2,B_1,B_0 -> 0$B_4$00, $B_1B_0B_3B_2$
  - If DBX = 1, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> 00$B_4$0, $B_2B_3B_0B_1$
  - If DBX = 1, DHX = 1: B_4,B_3,B_2,B_1,B_0 -> $B_4$000, $B_0B_1B_2B_3$

2. If source data width > destination data width, and if FIFO queuing and data unpacking is configured (PAM[1:0] = 1x):

- Padding with zero bytes.
- On the last read, only the number of LSB(s) up to the block size are written to the FIFO. The last written data to the destination is padded with zero bytes up to the destination data width. For example, if BNDT= 5, 32-bit reads, 16-bit writes:
  - If DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 0B_4, B_3B_2, B_1B_0
  - If DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$, -> $B_4$0, B_2B_3, B_0B_1

3. If source data width = destination data width (and BNDT[15:0] is not a multiple of that value):

- Padding with zero bytes. For example, if BNDT= 6, 32-bit reads, 32-bit writes are configured
  - If SBX = 0, DBX = 0, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $00B_5B_4$, $B_3B_2B_1B_0$
  - If SBX = 0, DBX = 0, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_5B_400, $B_1B_0B_3B_2$
  - If SBX = 0, DBX = 1, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $00B_4B_5$, $B_2B_3B_0B_1$
  - If SBX = 0, DBX = 1, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_4B_500, $B_0B_1B_2B_3$
  - If SBX = 1, DBX = 0, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $000B_4$, $B_3B_1B_2B_0$
  - If SBX = 1, DBX = 0, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $0B_400$, $B_2B_0B_3B_1$
  - If SBX = 1, DBX = 1, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $00B_40$, $B_1B_3B_0B_2$
  - If SBX = 1, DBX = 1, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_4000, $B_0B_2B_1B_3$

4. If source data width < destination data width, and if destination data is configured as right-aligned and, either configured with zero padding bytes (PAM[1:0] = 00), or configured with sign extension (PAM[1:0] = 01):

- Padding with zero bytes if PAM[1:0] = 00. For example, BNDT = 5, 8-bit reads, 32-bit writes, PAM[1:0] = 00:
  - If DBX =0, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> $000B_4$, $000B_3$, $000B_2$, $000B_1$, $000B_0$
- Padding with sign extension if PAM[1:0] = 01. For example, BNDT = 5, 8-bit reads, 32-bit writes, PAM[1:0] = 01:
  - If DBX = 0, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> $SSSB_4$, $SSSB_3$, $SSSB_2$, $SSSB_1$, $SSSB_0$

5. If source data width > destination data width, and if destination data is configured as right-aligned (PAM[1:0] = 00), or left-aligned (PAM[1:0] = 01):

- Padding with zero bytes. For example, BNDT = 5, 32-bit reads, 16-bit writes:
  - If SBX = 0, PAM[1:0] = 00, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $0B_4$, B_1B_0
  - If SBX = 0, PAM[1:0] = 00, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_40, B_0B_1
  - If SBX = 0, PAM[1:0] = 01, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_3B_2
  - If SBX = 0, PAM[1:0] = 01, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_2B_3
  - If SBX = 1, PAM[1:0] = 00, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $0B_4$, B_2B_0
  - If SBX = 1, PAM[1:0] = 00, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_40, B_0B_2
  - If SBX = 1, PAM[1:0] = 01, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_3B_1
  - If SBX = 0, PAM[1:0] = 01, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_1B_3

12.4.17 HPDMA privileged/unprivileged channel

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged agent via HPDMA_PRIVCFGR.PRIVx.

When a channel x is configured in a privileged state by a privileged agent, the following access control rules are applied:

- An unprivileged read access to a bitfield of this channel is forced to return 0, except for HPDMA_PRIVCFGR that is readable by an unprivileged agent.
- An unprivileged write access to a bitfield of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) state, the source and destination data transfers are privileged (respectively unprivileged) transfers over the AHB/AXI master port.

When a channel is configured in a privileged (or unprivileged) state and in linked-list mode, the loading of the next linked-list data structure from the HPDMA memory into its register file, is automatically performed with privileged (respectively unprivileged) transfers, via the HPDMA_CxCR.LAP allocated master port.

The HPDMA generates a privileged bus that reflects HPDMA_PRIVCFGR, to keep the other peripherals informed of the privileged/unprivileged state of each HPDMA channel x.

12.4.18 HPDMA error management

The HPDMA can manage and report to the user a transfer error, as follows, depending on the root cause.

Data transfer error

On a bus access (as an AHB/AXI single or a burst) to the source or the destination

- The source or destination target reports an AHB/AXI error.
- The programmed channel transfer is stopped (HPDMA_CxCR.EN cleared by the HPDMA hardware). The channel status register reports an idle state (HPDMA_CxSR.IDLEF = 1) and the data error (HPDMA_CxSR.DTEF = 1).
- After an HPDMA data transfer error, the user must perform a debug session, taking care of the product-defined memory mapping of the source and destination, including the protection attributes.
- After an HPDMA data transfer error, the user must issue a channel reset (set HPDMA_CxCR.RESET) to reset the hardware HPDMA channel data path and the FIFO content, before the user enables again the same channel for a next transfer.

Link transfer error

On a tentative update of a HPDMA channel register from the programmed LLI in the memory:

- The linked-list memory reports an AHB/AXI error.
- The programmed channel transfer is stopped (HPDMA_CxCR.EN cleared by the HPDMA hardware), the channel status register reports an idle state HPDMA_CxSR.IDLEF = 1), and the link error (HPDMA_CxSR.ULEF = 1).
- After an HPDMA link error, the user must perform a debug session, taking care of the product-defined memory mapping of the linked-list data structure (HPDMA_CxLBAR and HPDMA_CxLLR), including the protection attributes.
- After an HPDMA link error, the user must explicitly write the linked-list register file (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR, plus HPDMA_CxTR3 and HPDMA_CxBR2 when present), before the user enables again the same channel for a next transfer.

User setting error

On a tentative execution of an HPDMA transfer with an unauthorized user setting:

- The programmed channel transfer is disabled (HPDMA_CxCR.EN forced and cleared by the HPDMA hardware), preventing the next unauthorized programmed data transfer from being executed. The channel status register reports an idle state (HPDMA_CxSR.IDLEF = 1), and a user setting error (HPDMA_CxSR.USEF = 1).

- After an HPDMA user setting error, the user must perform a debug session, taking care of the HPDMA channel programming. A user setting error can be caused by one of the following:
  - A programmed null source block size without a programmed update of this value from the next LLI_1 (HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxLLR.UB1 = 0)
  - A programmed unaligned source start address being not a multiple of the programmed data width of a source burst transfer (HPDMA_CxSAR versus HPDMA_CxTR1.SDW_LOG2)
  - For a channel with 2D addressing, a programmed unaligned source address offset being not a multiple of the programmed data width of a source burst transfer (HPDMA_CxTR3.SAO versus HPDMA_CxTR1.SDW_LOG2)
  - For a channel with 2D addressing, a programmed unaligned block repeated source address offset being not a multiple of the programmed data width of a source burst transfer (HPDMA_CxBR2.BRSAO versus HPDMA_CxTR1.SDW_LOG2)
  - A programmed unaligned destination start address, being not a multiple of the programmed data width of a destination burst transfer (HPDMA_CxDAR versus HPDMA_CxTR1.DDW_LOG2)
  - For a channel with 2D addressing, a programmed unaligned destination address offset being not a multiple of the programmed data width of a destination burst transfer (HPDMA_CxTR3.DAO versus HPDMA_CxTR1.DDW_LOG2)
  - For a channel with 2D addressing, a programmed unaligned block repeated destination address offset being not a multiple of the programmed data width of a destination burst transfer (HPDMA_CxBR2.BRDAO versus HPDMA_CxTR1.DDW_LOG2)
  - A programmed source burst data width being superior than the data width of the allocated AXI/AHB source port (HPDMA_CxTR1.SDW_LOG2 and HPDMA_CxTR1.SAP vs the implemented AXI/AHB port data width).
  - A programmed destination burst data width being superior than the data width of the allocated AXI/AHB destination port (HPDMA_CxTR1.DDW_LOG2 and HPDMA_CxTR1.DAP vs the implemented AXI/AHB port data width).
  - A programmed linked-list item LLI_{n+1} with a null data transfer (HPDMA_CxLLR.UB1 = 1 and HPDMA_CxBR1. BNDT[15:0] = 0)

12.5 HPDMA in debug mode

When the device enters debug mode (core halted), any channel x can be individually either continued (default) or suspended, depending on the programmable control bit in the DBGMCU module.

Note: In debug mode, HPDMA_CxSR.SUSPF is not altered by a suspension from the programmable control bit in the DBGMCU module. In this case, HPDMA_CxSR.IDLEF can be checked to know the completion status of the channel suspension.

12.6 HPDMA in low-power modes

Table 90. Effect of low-power modes on HPDMA

Mode	Description
Sleep	No effect. HPDMA interrupts cause the device to exit Sleep mode.
Stop(1)	The content of HPDMA registers is kept when entering Stop mode.
Standby	The HPDMA is powered down, and must be reinitialized after exiting Standby mode.

1. Refer to *Section 12.3.2* to know if Stop mode is supported or not.

12.7 HPDMA interrupts

There is one HPDMA interrupt line for each channel, and separately for each CPU (if several ones in the devices).

Table 91. HPDMA interrupt requests

Interrupt acronym	Interrupt event	Interrupt enable	Event flag	Event clear method
HPDMA_CHx	Transfer complete	HPDMA_CxCR.TCIE	HPDMA_CxSR.TCF	Writes 1 to HPDMA_CxFCR.TCF
	Half transfer	HPDMA_CxCR.HTIE	HPDMA_CxSR.HTF	Writes 1 to HPDMA_CxFCR.HTF
	Data transfer error	HPDMA_CxCR.DTEIE	HPDMA_CxSR.DTEF	Writes 1 to HPDMA_CxFCR.DTEF
	Update link error	HPDMA_CxCR.ULEIE	HPDMA_CxSR.ULEF	Writes 1 to HPDMA_CxFCR.ULEF
	User setting error	HPDMA_CxCR.USEIE	HPDMA_CxSR.USEF	Writes 1 to HPDMA_CxFCR.USEF
	Suspended	HPDMA_CxCR.SUSPIE	HPDMA_CxSR.SUSPF	Writes 1 to HPDMA_CxFCR.SUSPF
	Trigger overrun	HPDMA_CxCR.TOFIE	HPDMA_CxSR.TOF	Writes 1 to HPDMA_CxFCR.TOF

An HPDMA channel x event can be:

- A transfer complete
- A half-transfer complete
- A transfer error, due to either:
  - A data transfer error
  - An update link error
  - A user setting error completed suspension
- A trigger overrun

Note: When a channel x transfer complete event occurs, the output signal hpdma_chx_tc is generated as a high pulse of one clock cycle.

An interrupt is generated following any xx event, provided that both:

- The corresponding interrupt event xx is enabled (HPDMA_CxCR.xxIE = 1)
- The corresponding event flag is cleared (HPDMA_CxSR.xxF = 0). This means that, after a previous same xx event occurrence, a software agent must have written 1 into the corresponding xx flag clear control bit (write 1 into HPDMA_CxFCR.xxF).

TCF (transfer complete) and HTF (half transfer) events generation is controlled by HPDMA_CxTR2.TCEM[1:0] as follows:

- A transfer-complete event is a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode HPDMA_CxTR2.TCEM[1:0].
- A half-transfer event is a half-block transfer, or a half 2D/repeated block transfer, depending on the transfer complete event mode HPDMA_CxTR2.TCEM[1:0].

  A half-block transfer occurs when half of the source block size bytes (rounded-up integer of HPDMA_CxBR1.BNDT[15:0] / 2) is transferred to the destination.

  A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded-up integer of (HPDMA_CxBR1.BRC[10:0] + 1) / 2) is transferred to the destination.

See *HPDMA channel x transfer register 2 (HPDMA_CxTR2)* for more details.

Note: The interrupt mode must be used (not the polling mode) to be notified on a half transfer when the write data transaction has been completed over the AXI destination allocated port (written at the destination memory-mapped address), and not just before when has been issued, at HPDMA level, this AXI burst transaction.

A transfer error rises in one of the following situations:

- During a single/burst data transfer from the source or to the destination (DTEF)
- During an update of an HPDMA channel register from the programmed LLI in memory (ULEF)
- During a tentative execution of an HPDMA channel with an unauthorized setting (USEF)

  The user must perform a debug session to correct the HPDMA channel programming versus the USEF root causes list (see *Section 12.4.18*).

A trigger overrun is described in *Trigger hit memorization and trigger overrun flag generation*.

12.8 HPDMA registers

The HPDMA registers must be accessed with an aligned 32-bit word data access.

12.8.1 HPDMA privileged configuration register (HPDMA_PRIVCFGR)

Address offset: 0x004

Reset value: 0x0000 0000

A write access to this register must be privileged. A read access can be privileged or unprivileged.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a HPDMA channel (HPDMA_CxCR.EN = 0), to individually allocate any channel x to the privileged or unprivileged world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PRIVx**: Privileged state of channel x (x = 15 to 0)
0: Unprivileged
1: Privileged

12.8.2 HPDMA configuration lock register (HPDMA_RCFGLOCKR)

Address offset: 0x008

Reset value: 0x0000 0000

This register can be written by a software agent with privileged attributes in order to individually lock, for example at boot time, the privileged attributes of any HPDMA channel/resource (to lock the setting of HPDMA_PRIVCFGR for any channel x at, for example at boot time).

A read access may be privileged or unprivileged.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **LOCKx**: Lock of the configuration of HPDMA_PRIVCFGR.PRIVx until a global HPDMA reset (x = 15 to 0)
This bit is cleared after reset and, once set, it cannot be reset until a global HPDMA reset.
0: privilege configuration of the channel x is writable.
1: privilege configuration of the channel x is not writable.

12.8.3 HPDMA masked interrupt status register (HPDMA_MISR)

Address offset: 0x0C

Reset value: 0x0000 0000

This is a read register.

This register contains the masked interrupt status bit MISx for each channel x. It is a logical OR of all the HPDMA_CxSR flags, each source flag being enabled by the corresponding HPDMA_CxCR interrupt enable bit.

Every bit is deasserted by hardware when writing 1 to the corresponding HPDMA_CxFCR flag clear bit.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel HPDMA_PRIVCFGR.PRIVx. A privileged software can read the full interrupt status. An unprivileged software is restricted to read the status of unprivileged channels, other privileged bitfields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MISx**: Masked interrupt status of channel x (x = 15 to 0)
0: No interrupt occurred on channel x.
1: An interrupt occurred on channel x.

12.8.4 HPDMA channel x linked-list base address register (HPDMA_CxLBAR)

Address offset: 0x050 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register must be written by a privileged software. It is either privileged readable or not, depending on the privileged state of the channel x HPDMA_PRIVCFGR.PRIVx.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This channel-based register is the linked-list base address of the memory region, for a given channel x, from which the LLIs describing the programmed sequence of the HPDMA transfers, are conditionally and automatically updated.

This 64-Kbyte aligned channel x linked-list base address is offset by the 16-bit HPDMA_CxLLR register that defines the word-aligned address offset for each LLI.

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

Bits 31:16 **LBA[31:16]**: Linked-list base address of HPDMA channel x

Bits 15:0 Reserved, must be kept at reset value.

12.8.5 HPDMA channel x flag clear register (HPDMA_CxFCR)

Address offset: 0x05C+ 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This is a write register privileged or unprivileged, depending on the privileged state of the channel x (HPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	w	w	w	w	w	w	w								

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **TOF**: Trigger overrun flag clear
0: No effect
1: Corresponding TOF flag cleared

Bit 13 **SUSPF**: Completed suspension flag clear
0: No effect
1: Corresponding SUSPF flag cleared

Bit 12 **USEF**: User setting error flag clear
0: No effect
1: Corresponding USEF flag cleared

Bit 11 **ULEF**: Update link transfer error flag clear
0: No effect
1: Corresponding ULEF flag cleared

Bit 10 **DTEF**: Data transfer error flag clear
0: No effect
1: Corresponding DTEF flag cleared

Bit 9 **HTF**: Half-transfer flag clear
0: No effect
1: Corresponding HTF flag cleared

Bit 8 **TCF**: Transfer complete flag clear
0: No effect
1: Corresponding TCF flag cleared

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

12.8.6 HPDMA channel x status register (HPDMA_CxSR)

Address offset: 0x060 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0001

This is a read register, reporting the channel status.

This register is privileged or unprivileged, depending on the privileged state of the channel (HPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[8:0]								
							r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	DRAINING F	IDLEF
	r	r	r	r	r	r	r							r	r

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 **FIFOL[8:0]**: Monitored FIFO level (in bytes)
Number of available bytes in the FIFO.

Note: After having suspended an active transfer, the user may need to read FIFOL[8:0], additionally to HPDMA_CxBR1.BDNT[15:0] and HPDMA_CxBR1.BRC[10:0], to know how many data have been transferred to the destination. Before reading, the user may wait for the transfer to be suspended (HPDMA_CxSR.SUSPF = 1).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOF**: Trigger overrun flag
0: No trigger overrun event
1: A trigger overrun event occurred.

Bit 13 **SUSPF**: Completed suspension flag
0: No completed suspension event
1: A completed suspension event occurred.

Bit 12 **USEF**: User setting error flag
0: No user setting error event
1: A user setting error event occurred.

Bit 11 **ULEF**: Update link transfer error flag
0: No update link transfer error event
1: A master bus error event occurred while updating a linked-list register from memory.

Bit 10 **DTEF**: Data transfer error flag
0: No data transfer error event
1: A master bus error event occurred on a data transfer.

Bit 9 **HTF**: Half-transfer flag
0: No half-transfer event
1: A half-transfer event occurred.
A half-transfer event is either a half-block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode (HPDMA_CxTR2.TCEM[1:0]).
A half-block transfer occurs when half of the bytes of the source block size (rounded up integer of HPDMA_CxBR1.BNDT[15:0]/2) has been transferred to the destination.
A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded up integer of (HPDMA_CxBR1.BRC[10:0]+1)/2)) has been transferred to the destination.

Bit 8 **TCF**: Transfer complete flag
0: No transfer complete event
1: A transfer complete event occurred.
A transfer complete event is either a block transfer complete, a 2D/repeated block transfer complete, a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode (HPDMA_CxTR2.TCEM[1:0]).

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

Bit 1 **DRAININGF**: FIFO draining flag
0: No received order to drain the FIFO or channel FIFO draining completed
1: Ongoing channel FIFO draining and channel termination
This flag is asserted after that the order to drain the FIFO is received, and meanwhile that the channel is enabled and draining the FIFO and before the channel is terminated with a drained FIFO.
This flag is deasserted by hardware when the enabled channel has been ordered to be terminated with FIFO draining, and the hardware effectively terminates.

Bit 0 **IDLEF**: Idle flag
0: Channel not in idle state
1: Channel in idle state
This idle flag is deasserted by hardware when the channel is enabled (HPDMA_CxCR.EN = 1) with a valid channel configuration (no USEF to be immediately reported).
This idle flag is asserted after hard reset or by hardware when the channel is back in idle state (in suspended or disabled state).

12.8.7 HPDMA channel x control register (HPDMA_CxCR)

Address offset: 0x064 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of the channel x (HPDMA_PRIVCFGR.PRIVx).

This register is used to control a channel (activate, suspend, abort or disable it).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	LAP	LSM
								rw	rw					rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	DRAIN FIFO	SUSP	RESET	EN
	rw	rw	rw	rw	rw	rw	rw					w	rw	w	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **PRIO[1:0]:** Priority level of the channel x HPDMA transfer versus others
00: Low priority, low weight
01: Low priority, mid weight
10: Low priority, high weight
11: High priority
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 21:18 Reserved, must be kept at reset value.

Bit 17 **LAP**: Linked-list allocated port
This bit is used to allocate the master port for the update of the HPDMA linked-list registers from the memory.
0: Port 0 (AXI) allocated
1: Port 1 (AHB) allocated
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 16 **LSM**: Link step mode
0: Channel executed for the full linked-list and completed at the end of the last LLI (HPDMA_CxLLR = 0). The 16 low-significant bits of the link address are null (LA[15:0] = 0) and all the update bits are null (UT1 = UB1 = UT2 = USA = UDA = ULL = 0 and UT3 = UB2 = 0 if present). Then HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0 if present.
1: Channel executed once for the current LLI
First the (possible 1D/repeated) block transfer is executed as defined by the current internal register file until HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0, if present. Secondly the next linked-list data structure is conditionally uploaded from memory as defined by HPDMA_CxLLR. Then channel execution is completed.
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOIE**: Trigger overrun interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **SUSPIE**: Completed suspension interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **USEIE**: User setting error interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **ULEIE**: Update link transfer error interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **DTEIE**: Data transfer error interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 9 **HTIE**: Half-transfer complete interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 8 **TCIE**: Transfer complete interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **DRAINFIFO:** Empty FIFO
This bit is write only. Writing 0 has no impact.
Writing 1 to RESET in this register causes the hardware to deassert this DRAINFIFO bit, whatever is written into this DRAINFIFO. Else:
Software must write 1 in order to drain the channel FIFO on a suspended channel.
If the software concurrently writes 1 to DRAINFIFO and 1 to SUSP and 0 to reset, the hardware suspends the channel and drains the FIFO by popping out the residual FIFO bytes to the destination (see the programming sequence in *Figure 83*).
0: Write: no action
1: Write: drain FIFO after having suspended the channel

Bit 2 **SUSP:** Suspend
Writing 1 to RESET in this register causes the hardware to deassert this SUSP bit, whatever is written into this SUSP. Else:
Software must write 1 in order to suspend an active channel (a channel with an ongoing HPDMA transfer over its master ports).
The software must write 0 in order to resume a suspended channel, following the programming sequence detailed in *Figure 81*.
0: Write: resume channel, read: channel not suspended
1: Write: suspend channel, read: channel suspended

Bit 1 **RESET:** Reset

This bit is write only. Writing 0 has no impact. Writing 1 implies the reset of the following: the FIFO, the channel internal state, SUSP, DRAINFIFO and EN bits (whatever is written receptively in bit 3 and bit 2 and bit 0).

The reset is effective when the channel is in steady state, meaning one of the following:

- active channel in suspended state (HPDMA_CxSR.SUSPF = 1 and HPDMA_CxSR.IDLEF = HPDMA_CxCR.EN = 1 and HPDMA_CxSR.DRAININGF = 0)
- channel in disabled state (HPDMA_CxSR.IDLEF = 1 and HPDMA_CxCR.EN = 0).

After writing a RESET, to continue using this channel, the user must explicitly reconfigure the channel including the hardware-modified configuration registers (HPDMA_CxBR1, HPDMA_CxSAR and HPDMA_CxDAR) before enabling again the channel (see the programming sequence in *Figure 82*).

0: No channel reset

1: Channel reset

Bit 0 **EN:** Enable

Writing 1 to RESET in this register causes the hardware to deassert this EN bit, whatever is written into this bit 0. Else:

This bit is deasserted by hardware when there is a transfer error (master bus error or user setting error) or when there is a channel transfer complete (channel ready to be configured, for example if LSM = 1 at the end of a single execution of the LLI).

Else, this bit can be asserted by software.

Writing 0 into this EN bit is ignored.

0: Write: ignored, read: channel disabled

1: Write: enable channel, read: channel enabled

12.8.8 HPDMA channel x transfer register 1 (HPDMA_CxTR1)

Address offset: 0x090 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of the channel x in HPDMA_PRIVCFGR.PRIVx.

This register controls the transfer of a channel x.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed. Then the hardware has de-asserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by HPDMA from the memory if HPDMA_CxLLR.UT1 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DAP	Res.	DWX	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2[1:0]	
	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2[1:0]	
	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DAP**: Destination allocated port

This bit is used to allocate the master port for the destination transfer

0: Port 0 (AXI) allocated

1: Port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DWX**: Destination word exchange

If the destination data size is not a double-word, this bit is ignored.

If the destination data size is a double-word and if destination bus is AXI (DAP = 0):

0: No word-based exchanged within double-word

1: The two consecutive (post PAM) words are exchanged in each destination double-word.

Bit 27 **DHX**: Destination half-word exchange

If the destination data size is shorter than a word, this bit is ignored.

If the destination data size is a word or double-word and if destination bus is AXI (DAP = 0):

0: No half-word-based exchanged within word

1: The two consecutive (post PAM) half-words are exchanged in each destination word.

Bit 26 **DBX**: Destination byte exchange

If the destination data size is a byte, this bit is ignored.

If the destination data size is not a byte:

0: No byte-based exchange within half-word

1: The two consecutive (post PAM) bytes are exchanged in each destination half-word.

Bits 25:20 **DBL_1[5:0]**: Destination burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If DBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width DDW_LOG2[1:0].

Note: If a burst transfer crossed a 1- or 4-Kbyte address boundary on respectively an AHB or AXI transfer, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB/AXI protocol.

If the burst length exceeds 16 on an AHB transfer, or if the burst on an AXI transfer is both with fixed addressing (DINC = 0) and with a burst length which exceeds 16, the HPDMA modifies and shortens the programmed burst into bursts of lower length, to be compliant with the AHB or AXI protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with HPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 19 **DINC**: Destination incrementing burst

0: Fixed burst

1: Contiguously incremented burst

The destination address, pointed by HPDMA_CxDAR, is kept constant after a burst beat/single transfer, or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DDW_LOG2[1:0]**: Binary logarithm of the destination data width of a burst, in bytes
00: Byte
01: Half-word (2 bytes)
10: Word (4 bytes)
11: Double-word (8 bytes)

Note: A destination burst data width must be less or equal to the implemented data width of the allocated AXI/AHB destination port via the DAP bit (it is recommended to be equal for best performance). Otherwise a user setting error is reported and no transfer is issued.

A destination burst transfer must have an aligned destination i) address (HPDMA_CxDAR[2:0]), ii) address offset HPDMA_CxTR3.DAO[2:0] if present, with its destination data width (DDW_LOG2[2:0]). Otherwise a user setting error is reported and no transfer is issued.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SAP**: Source allocated port
This bit is used to allocate the master port for the source transfer
0: Port 0 (AXI) allocated
1: Port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 13 **SBX**: Source byte exchange within the unaligned half-word of each source word
If the source data width is shorter than a word, this bit is ignored.
If the source data width is a word or a double-word, and if source bus is AXI (SAP = 0):
0: No byte-based exchange within the unaligned half-word of each source word
1: The two consecutive bytes within the unaligned half-word of each source word are exchanged.

Bits 12:11 **PAM[1:0]**: Padding/alignment mode
If DDW_LOG2[1:0] = SDW_LOG2[1:0]: if the data width of a burst destination transfer is equal to the data width of a burst source transfer, these bits are ignored.
Else, in the following enumerated values, the condition PAM_1 is when destination data width is higher than source data width, and the condition PAM_2 is when source data width is higher than destination data width.
Condition: PAM_1
00: Source data is transferred as right aligned, padded with 0s up to the destination data width
01: Source data is transferred as right aligned, sign extended up to the destination data width
10-11: Successive source data are FIFO queued and packed at the destination data width, in a left (LSB) to right (MSB) order (named little endian), before a destination transfer
Condition: PAM_2
00: Source data is transferred as right aligned, left-truncated down to the destination data width
01: Source data is transferred as left-aligned, right-truncated down to the destination data width
10-11: Source data is FIFO queued and unpacked at the destination data width, to be transferred in a left (LSB) to right (MSB) order (named little endian) to the destination

Note: If the transfer from the source peripheral is configured with peripheral flow-control mode (SWREQ = 0 and PFREQ = 1 and DREQ = 0), and if the destination data width > the source data width, packing is not supported.

Bit 10 Reserved, must be kept at reset value.

Bits 9:4 **SBL_1[5:0]**: Source burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If SBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width SDW_LOG2[1:0].

Note: If a burst transfer crossed a 1- or 4-Kbyte address boundary on respectively an AHB or an AXI transfer, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If the burst length exceeds 16 on a AHB transfer, or if the burst on a AXI transfer is both with fixed addressing (SINC = 0) and with a burst length which exceeds 16, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with HPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 3 **SINC**: Source incrementing burst

0: Fixed burst

1: Contiguously incremented burst

The source address, pointed by HPDMA_CxSAR, is kept constant after a burst beat/single transfer or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SDW_LOG2[1:0]**: Binary logarithm of the source data width of a burst in bytes

00: Byte

01: Half-word (2 bytes)

10: Word (4 bytes)

11: Double-word (8 bytes)

Note: A source burst data width must be less or equal to the implemented data width of the allocated AXI/AHB source port via the SAP bit (it is recommended to be equal for best performance). Otherwise a user setting error is reported and no transfer is issued.

A source burst transfer must have an aligned source i) address (HPDMA_CxSAR[2:0]), ii) address offset HPDMA_CxTR3.SAO[2:0] if present, with its source data width (SDW_LOG2[1:0]). Otherwise a user setting error is reported and no transfer is issued.

12.8.9 HPDMA channel x transfer register 2 (HPDMA_CxTR2)

Address offset: 0x094 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (the hardware deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by HPDMA from the memory, if HPDMA_CxLLR.UT2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	Res.	TRIGSEL[5:0]					
rw	rw					rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGM[1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	REQSEL[7:0]							
rw	rw		rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **TCEM[1:0]**: Transfer complete event mode

These bits define the transfer granularity for the transfer complete and half-transfer complete events generation.

00: At block level (when HPDMA_CxBR1.BNDT[15:0] = 0): The complete (and the half) transfer event is generated at the (respectively half of the) end of a block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with HPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half-transfer event is generated.

01: if the channel is not 2D addressing capable: same as 00; if the channel is 2D addressing capable: at 2D/repeated block level, meaning when HPDMA_CxBR1.BRC[10:0] = 0 and HPDMA_CxBR1.BNDT[15:0] = 0), the complete (and the half) transfer event is generated at the end (respectively half of the end) of the 2D/repeated block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with HPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half-transfer event is generated.

10: At LLI level: The complete transfer event is generated at the end of the LLI transfer, including the update of the LLI if any. The half-transfer event is generated at the half of the LLI data transfer (the LLI data transfer being a block transfer or a 2D/repeated block transfer for a 2D addressing capable channel), if any data transfer.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with HPDMA_CxBR1.BNDT[15:0] = 0), then the half-transfer event is not generated, and the transfer complete event is generated when is completed the loading of the LLI_1.

11: At channel level: The complete transfer event is generated at the end of the last LLI transfer. The half-transfer event is generated at the half of the data transfer of the last LLI. The last LLI updates the link address HPDMA_CxLLR.LA[15:2] to zero and clears all the HPDMA_CxLLR update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UT3 and UB2 if present). If the channel transfer is continuous/infinite, no event is generated.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **TRIGPOL[1:0]**: Trigger event polarity

These bits define the polarity of the selected trigger event input defined by TRIGSEL[5:0].

00: No trigger (masked trigger event)

01: Trigger on the rising edge

10: Trigger on the falling edge

11: Same as 00

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **TRIGSEL[5:0]**: Trigger event input selection

These bits select the trigger event input of the HPDMA transfer (as per *Section 12.3.6*), with an active trigger event if TRIGPOL[1:0] ≠ 00.

Bits 15:14 **TRIGM[1:0]**: Trigger mode

These bits define the transfer granularity for its conditioning by the trigger.

If the channel x is enabled (HPDMA_CxCR.EN asserted) with TRIGPOL[1:0] = 00 or 11, these TRIGM[1:0] bits are ignored.

Else, an HPDMA transfer is conditioned by at least one trigger hit:

00: at block level: the first burst read of each block transfer is conditioned by one hit trigger. This is also valid for a 2D addressing capable channel: for each block even if a 2D/repeated block is configured with HPDMA_CxBR1.BRC[10:0] ≠ 0.

01: if the channel is not 2D addressing capable: same as 00; if the channel is 2D addressing capable: at 2D/repeated block level, meaning the first burst read of a 2D/repeated block transfer is conditioned by one hit trigger.

10: at link level: a LLI link transfer is conditioned by one hit trigger. The LLI data transfer (if any) is not conditioned.

11: at programmed burst level: If SWREQ = 1, each programmed burst read is conditioned by one hit trigger. If SWREQ = 0, each programmed burst that is requested by the selected peripheral, is conditioned by one hit trigger.

– If the peripheral is programmed as a source (DREQ = 0) of the LLI data transfer, each programmed burst read is conditioned.

– If the peripheral is programmed as a destination (DREQ = 1) of the LLI data transfer, each programmed burst write is conditioned. The first memory burst read of a (possibly 2D/repeated) block, also named as the first ready FIFO-based source burst, is gated by the occurrence of both the hardware request and the first trigger hit.

The HPDMA monitoring of a trigger for channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (TRIGPOL[1:0] = 01 or respectively TRIGPOL[1:0] = 10).

The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer; and if a new trigger is detected then, this hit is internally memorized to grant the next transfer, as long as the defined rising or falling edge is not modified, and the TRIGSEL[5:0] is not modified, and the channel is enabled.

Transferring a next LLI_{n+1} that updates the HPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the memorized hit of the formerly defined LLI_n trigger.

After a first new trigger hit_{n+1} is memorized, if another second trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (HPDMA_CxSR.TOF = 1), and an interrupt is generated if enabled (HPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Note: When the source block size is not a multiple of the source burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, if TRIGM[1:0] = 11 and (SWREQ = 1 or (SWREQ = 0 and DREQ = 0)), the shortened burst transfer (by singles or/and by bursts of lower length) is conditioned once by the trigger.

When the programmed destination burst is internally shortened by singles or/and by bursts of lower length (versus FIFO size, versus block size, 1/4-Kbyte boundary address crossing maximum burst length versus AHB/AXI protocol): if the trigger is conditioning the programmed destination burst (if TRIGM[1:0] = 11 and SWREQ = 0 and DREQ = 1), this shortened destination burst transfer is conditioned once by the trigger.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PFREQ**: Hardware request in peripheral flow control mode

Important: If a given channel x is not implemented with this feature, this bit is reserved and PFREQ is not present (see Section 12.3.5 for the list of the implemented channels with this feature.

If the channel x is activated (HPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: The selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in HPDMA control mode. The HPDMA is programmed with HPDMA_CxCBR1.BNDT[15:0] and this is internally used by the hardware for the block transfer completion.

1: The selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in peripheral control mode. The HPDMA block transfer can be early completed by the peripheral itself (see *Section 12.3.5* for more details).

Note: In peripheral flow control mode, there are the following restrictions:

- no 2D/repeated block support (HPDMA_CxBR1.BRC[10:0] must be set to 0 if present)

- the peripheral must be set as the source of the transfer (DREQ = 0).

- data packing to a wider destination width is not supported (if destination width > source data width, HPDMA_CxTR1.PAM[1] must be set to 0).

Bit 11 **BREQ**: Block hardware request

If the channel x is activated (HPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: The selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a burst level.

1: The selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a block level (see *Section 12.3.4*).

Bit 10 **DREQ**: Destination hardware request

This bit is ignored if channel x is activated (HPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else:

0: Selected hardware request driven by a source peripheral (request signal taken into account by the HPDMA transfer scheduler over the source/read port)

1: Selected hardware request driven by a destination peripheral (request signal taken into account by the HPDMA transfer scheduler over the destination/write port)

Note: If the channel x is activated (HPDMA_CxCR.EN is asserted) with SWREQ = 0 and PFREQ = 1 (peripheral hardware request with peripheral flow-control mode), any software assertion to this DREQ bit is ignored: in peripheral flow-control mode, only a peripheral-to-memory transfer is supported.

Bit 9 **SWREQ**: Software request

This bit is internally taken into account when HPDMA_CxCR.EN is asserted.

0: No software request. The selected hardware request REQSEL[7:0] is taken into account.

1: Software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[7:0] is ignored.

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **REQSEL[7:0]**: Hardware request selection

These bits are ignored if channel x is activated (HPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else, the selected hardware request is internally taken into account as per *Section 12.3.3*.

Caution: The user must not assign a same input hardware request (same REQSEL[7:0] value) to different active HPDMA channels (HPDMA_CxCR.EN = 1 and HPDMA_CxTR2.SWREQ = 0 for these channels). The HPDMA is not intended to hardware support the case of simultaneous enabled channels incorrectly configured with a same hardware peripheral request signal, and there is no user setting error reporting.

12.8.10 HPDMA channel x block register 1 (HPDMA_CxBR1)

Address offset: 0x098 + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has de-asserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer:

- If HPDMA_CxLLR.UB1 = 1, this register is automatically updated by the HPDMA from the next LLI in memory.
- If HPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the bitfield BNDT[15:0].
- If all the update bits HPDMA_CxLLR.Uxx are null and if HPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for BNDT[15:0] after each execution of this final LLI
- If HPDMA_CxLLR = 0, this register and BNDT[15:0] are kept as null, channel x is completed.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BNDT[15:0]**: Block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this bitfield becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if HPDMA_CxLLR.UB1 = 1, this bitfield is updated with the LLI in the memory.
- if HPDMA_CxLLR.UB1 = 0 and if there is at least one non null Uxx update bit, this bitfield is internally restored to the programmed value.
- if all HPDMA_CxLLR.Uxx = 0 and if HPDMA_CxLLR.LA[15:0] = 0, this bitfield is internally restored to the programmed value (infinite/continuous last LLI).
  - if HPDMA_CxLLR = 0, this bitfield is kept as zero following the last LLI data transfer.

12.8.11 HPDMA channel x alternate block register 1 (HPDMA_CxBR1)

Address offset: 0x098 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has de-asserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer:

- if HPDMA_CxLLR.UB1 = 1, this register is automatically updated by the HPDMA from the next LLI in memory.
- If HPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the bitfields BNDT[15:0] and BRC[10:0].
- If all the update bits HPDMA_CxLLR.Uxx are null and if HPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for the bitfields BNDT[15:0] and BRC[10:0] after each execution of this final LLI
- If HPDMA_CxLLR = 0, BNDT[15:0] and BRC[10:0] are kept as null, channel x is completed.

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

Bit 31 **BRDDEC**: Block repeat destination address decrement

0: At the end of a block transfer, the HPDMA_CxDAR register is updated by adding the programmed offset HPDMA_CxBR2.BRDAO to the current HPDMA_CxDAR value (current destination address)

1: At the end of a block transfer, the HPDMA_CxDAR register is updated by subtracting the programmed offset HPDMA_CxBR2.BRDAO from the current HPDMA_CxDAR value (current destination address)

Note: On top of this increment/decrement (depending on BRDDEC), HPDMA_CxDAR is in the same time also updated by the increment/decrement (depending on DDEC) of the HPDMA_CxTR3.DAO value, as it is usually done at the end of each programmed burst transfer.

Bit 30 **BRSDEC**: Block repeat source address decrement

0: At the end of a block transfer, the HPDMA_CxSAR register is updated by adding the programmed offset HPDMA_CxBR2.BRSAO to the current HPDMA_CxSAR value (current source address)

1: At the end of a block transfer, the HPDMA_CxSAR register is updated by subtracting the programmed offset HPDMA_CxBR2.BRSAO from the current HPDMA_CxSAR value (current source address)

Note: On top of this increment/decrement (depending on BRSDEC), HPDMA_CxSAR is in the same time also updated by the increment/decrement (depending on SDEC) of the HPDMA_CxTR3.SAO value, as it is done after any programmed burst transfer.

Bit 29 **DDEC**: destination address decrement

0: At the end of a programmed burst transfer to the destination, the HPDMA_CxDAR register is updated by adding the programmed offset HPDMA_CxTR3.DAO to the current HPDMA_CxDAR value (current destination address)

1: At the end of a programmed burst transfer to the destination, the HPDMA_CxDAR register is updated by subtracting the programmed offset HPDMA_CxTR3.DAO to the current HPDMA_CxDAR value (current destination address)

Bit 28 **SDEC**: source address decrement

0: At the end of a programmed burst transfer from the source, the HPDMA_CxSAR register is updated by adding the programmed offset HPDMA_CxTR3.SAO to the current HPDMA_CxSAR value (current source address)

1: At the end of a programmed burst transfer from the source, the HPDMA_CxSAR register is updated by subtracting the programmed offset HPDMA_CxTR3.SAO to the current HPDMA_CxSAR value (current source address)

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **BRC[10:0]**: Block repeat counter

This bitfield contains the number of repetitions of the current block (0 to 2047).

When the channel is enabled, this bitfield becomes read-only. After decrements, this bitfield indicates the remaining number of blocks, excluding the current one. This counter is hardware decremented for each completed block transfer.

Once the last block transfer is completed (BRC[10:0] = BNDT[15:0] = 0):

- If HPDMA_CxLLR.UB1 = 1, all HPDMA_CxBR1 bitfields are updated by the next LLI in the memory.
- If HPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this bitfield is internally restored to the programmed value.
- If all HPDMA_CxLLR.Uxx = 0 and if HPDMA_CxLLR.LA[15:0] ≠ 0, this bitfield is internally restored to the programmed value (infinite/continuous last LLI).
- If HPDMA_CxLLR = 0, this bitfield is kept as zero following the last LLI and data transfer.

Bits 15:0 **BNDT[15:0]**: Block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this bitfield becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred.

BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- If HPDMA_CxLLR.UB1 = 1, this bitfield is updated with the LLI in the memory.
- If HPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this bitfield is internally restored to the programmed value.
- If all HPDMA_CxLLR.Uxx = 0 and if HPDMA_CxLLR.LA[15:0] ≠ 0, this bitfield is internally restored to the programmed value (infinite/continuous last LLI).

If HPDMA_CxLLR = 0, this bitfield is kept as zero following the last LLI data transfer.

12.8.12 HPDMA channel x source address register (HPDMA_CxSAR)

Address offset: 0x09C + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register configures the source start address of a transfer.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer from the source.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.USA = 1.

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

Bits 31:0 **SA[31:0]**: Source address

This bitfield is the pointer to the address from which the next data is read.

During the channel activity, depending on the source addressing mode (HPDMA_CxTR1.SINC), this bitfield is kept fixed or incremented by the data width (HPDMA_CxTR1.SDW_LOG2[1:0]) after each burst source data, reflecting the next address from which data is read.

During the channel activity, this address is updated after each completed source burst, consequently to:

- The programmed source burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (HPDMA_CxTR1.SINC = 1), then the additional address offset value is the programmed burst size, as defined by HPDMA_CxTR1.SBL_1[5:0] and HPDMA_CxTR1.SDW_LOG2[1:0]
- The additional source incremented/decremented offset value as programmed by HPDMA_CxBR1.SDEC and HPDMA_CxTR3.SAO[12:0]
- Once/if completed source block transfer, for a channel x with 2D addressing capability (x = 12 to 15). additional block repeat source incremented/decremented offset value as programmed by HPDMA_CxBR1.BRSDEC and HPDMA_CxBR2.BRSAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by HPDMA from the memory, provided the LLI is set with HPDMA_CxLLR.USA = 1.

Note: A source address must be aligned with the programmed data width of a source burst (SA versus HPDMA_CxTR1.SDW_LOG2). Else, a user setting error is reported and no transfer is issued.

When the source block size is not a multiple of the source burst size and is a multiple of the source data width, the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional HPDMA_CxTR3.SAO[12:0] is not applied.

12.8.13 HPDMA channel x destination address register (HPDMA_CxDAR)

Address offset: 0x0A0 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register configures the destination start address of a transfer.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer to the destination.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by HPDMA from the memory if HPDMA_CxLLR.UDA = 1.

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

Bits 31:0 **DA[31:0]**: destination address

This bitfield is the pointer to the address from which the next data is written.

During the channel activity, depending on the destination addressing mode (HPDMA_CxTR1.DINC), this bitfield is kept fixed or incremented by the data width (HPDMA_CxTR1.DDW_LOG2[1:0]) after each burst destination data, reflecting the next address from which data is written.

During the channel activity, this address is updated after each completed destination burst, consequently to:

– the programmed destination burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (HPDMA_CxTR1.DINC = 1), then the additional address offset value is the programmed burst size, as defined by HPDMA_CxTR1.DBL_1[5:0] and HPDMA_CxTR1.DDW_LOG2[1:0]

– the additional destination incremented/decremented offset value as programmed by HPDMA_CxBR1.DDEC and HPDMA_CxTR3.DAO[12:0]

– once/if completed destination block transfer, for a channel x with 2D addressing capability , the additional block repeat destination incremented/decremented offset value as programmed by HPDMA_CxBR1.BRDDEC and HPDMA_CxBR2.BRDAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by the HPDMA from the memory, provided the LLI is set with HPDMA_CxLLR.UDA = 1.

Note: A destination address must be aligned with the programmed data width of a destination burst (DA versus HPDMA_CxTR1.DDW_LOG2). Else, a user setting error is reported and no transfer is issued.

12.8.14 HPDMA channel x transfer register 3 (HPDMA_CxTR3)

Address offset: 0x0A4 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.UT3 = 1.

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **DAO[12:0]**: Destination address offset increment

The destination address, pointed by HPDMA_CxDAR, is incremented or decremented (depending on HPDMA_CxBR1.DDEC) by this offset DAO[12:0] for each programmed destination burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (HPDMA_CxTR1.DINC = 1).

Note: A destination address offset must be aligned with the programmed data width of a destination burst (DAO versus HPDMA_CxTR1.DDW_LOG2). Else, a user setting error is reported and no transfer is issued.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **SAO[12:0]**: Source address offset increment

The source address, pointed by HPDMA_CxSAR, is incremented or decremented (depending on HPDMA_CxBR1.SDEC) by this offset SAO[12:0] for each programmed source burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (HPDMA_CxTR1.SINC = 1).

Note: A source address offset must be aligned with the programmed data width of a source burst (SAO versus HPDMA_CxTR1.SDW_LOG2). Else a user setting error is reported and none transfer is issued.

When the source block size is not a multiple of the destination burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional HPDMA_CxTR3.SAO[12:0] is not applied.

12.8.15 HPDMA channel x block register 2 (HPDMA_CxBR2)

Address offset: 0x0A8 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a 2D/repeated block level.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.UB2 = 1.

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

Bits 31:16 **BRDAO[15:0]**: Block repeated destination address offset

For a channel with 2D addressing capability, this bitfield is used to update (by addition or subtraction depending on HPDMA_CxBR1.BRDDEC) the current destination address (HPDMA_CxDAR) at the end of a block transfer.

Note: A block repeated destination address offset must be aligned with the programmed data width of a destination burst (BRDAO versus HPDMA_CxTR1.DDW_LOG2). Else a user setting error is reported and no transfer is issued.
BRDAO[15:0] must be set to 0 in peripheral flow-control mode (if HPDMA_CxTR2.PFREQ = 1).

Bits 15:0 **BRSAO[15:0]**: Block repeated source address offset

For a channel with 2D addressing capability, this bitfield is used to update (by addition or subtraction depending on HPDMA_CxBR1.BRSDEC) the current source address (HPDMA_CxSAR) at the end of a block transfer.

Note: A block repeated source address offset must be aligned with the programmed data width of a source burst (BRSAO versus HPDMA_CxTR1.SDW_LOG2). Else a user setting error is reported and no transfer is issued.
BRSAO[15:0] must be set to 0 in peripheral flow-control mode (if HPDMA_CxTR2.PFREQ = 1).

12.8.16 HPDMA channel x linked-list address register (HPDMA_CxLLR)

Address offset: 0x0CC + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw											rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update HPDMA_CxTR1 from memory

This bit controls the update of HPDMA_CxTR1 from the memory during the link transfer.
0: No HPDMA_CxTR1 update
1: HPDMA_CxTR1 update

Bit 30 **UT2**: Update HPDMA_CxTR2 from memory

This bit controls the update of HPDMA_CxTR2 from the memory during the link transfer.
0: No HPDMA_CxTR2 update
1: HPDMA_CxTR2 update

Bit 29 **UB1**: Update HPDMA_CxBR1 from memory

This bit controls the update of HPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if HPDMA_CxLLR ≠ 0, the linked-list is not completed.
HPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.
0: No HPDMA_CxBR1 update from memory (HPDMA_CxBR1.BNDT[15:0] restored if any link transfer)
1: HPDMA_CxBR1 update

Bit 28 **USA**: Update HPDMA_CxSAR from memory

This bit controls the update of HPDMA_CxSAR from the memory during the link transfer.
0: No HPDMA_CxSAR update
1: HPDMA_CxSAR update

Bit 27 **UDA**: Update HPDMA_CxDAR register from memory
This bit is used to control the update of HPDMA_CxDAR from the memory during the link transfer.
0: No HPDMA_CxDAR update
1: HPDMA_CxDAR update

Bits 26:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update HPDMA_CxLLR register from memory
This bit is used to control the update of HPDMA_CxLLR from the memory during the link transfer.
0: No HPDMA_CxLLR update
1: HPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: Pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list HPDMA register file.
Else, this bitfield is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list HPDMA internal register file (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR).

Caution: the user must program this pointer not to exceed the 64-Kbyte addressable space defined by the link base address register HPDMA_CxLBAR. The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

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

12.8.17 HPDMA channel x alternate linked-list address register (HPDMA_CxLLR)

Address offset: 0x0CC + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw	rw	rw									rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update HPDMA_CxTR1 from memory

This bit controls the update of HPDMA_CxTR1 from the memory during the link transfer.

0: No HPDMA_CxTR1 update

1: HPDMA_CxTR1 update

Bit 30 **UT2**: Update HPDMA_CxTR2 from memory

This bit controls the update of HPDMA_CxTR2 from the memory during the link transfer.

0: No HPDMA_CxTR2 update

1: HPDMA_CxTR2 update

Bit 29 **UB1**: Update HPDMA_CxBR1 from memory

This bit controls the update of HPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if HPDMA_CxLLR ≠ 0, the linked-list is not completed.
HPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.

0: No HPDMA_CxBR1 update from memory (HPDMA_CxBR1.BNDT[15:0] restored if any link transfer)

1: HPDMA_CxBR1 update

Bit 28 **USA**: Update HPDMA_CxSAR from memory

This bit controls the update of HPDMA_CxSAR from the memory during the link transfer.

0: No HPDMA_CxSAR update

1: HPDMA_CxSAR update

Bit 27 **UDA**: Update HPDMA_CxDAR register from memory
This bit is used to control the update of HPDMA_CxDAR from the memory during the link transfer.
0:No HPDMA_CxDAR update
1: HPDMA_CxDAR update

Bit 26 **UT3**: Update HPDMA_CxTR3 from memory
This bit controls the update of HPDMA_CxTR3 from the memory during the link transfer.
0: No HPDMA_CxTR3 update
1: HPDMA_CxTR3 update

Bit 25 **UB2**: Update HPDMA_CxBR2 from memory
This bit controls the update of HPDMA_CxBR2 from the memory during the link transfer.
0: No HPDMA_CxBR2 update
1: HPDMA_CxBR2 update

Bits 24:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update HPDMA_CxLLR register from memory
This bit is used to control the update of HPDMA_CxLLR from the memory during the link transfer.
0: No HPDMA_CxLLR update
1: HPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: Pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL= 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list HPDMA register file.
Else, this bitfield is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list HPDMA internal register file (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR and HPDMA_CxLLR).

Caution: the user must program this pointer not to exceed the 64-Kbyte addressable space defined by the link base address register HPDMA_CxLBAR. The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

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

12.8.18 HPDMA register map

Table 92. HPDMA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	HPDMA_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	HPDMA_ RCFGLOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 92. HPDMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00C	HPDMA_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010 - 0x04C	Reserved																																
0x050 + 0x80 * x (x = 0 to 15)	HPDMA_CxLBAR	LBA[31:16]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x05C + 0x080 * x (x=0 to 15)	HPDMA_CxFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0	0	0	0	0	0	0								
0x060 + 0x080 * x (x=0 to 15)	HPDMA_CxSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[8:0]									Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	DRAININGF	IDLEF
	Reset value								0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0							0	1
0x064 + 0x080 * x (x=0 to 15)	HPDMA_CxCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	LAP	LSM	Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	DRAINFIFO	SUSP	RESET	EN
	Reset value									0	0					0	0		0	0	0	0	0	0	0					0	0	0	0
0x090 + 0x080 * x (x=0 to 15)	HPDMA_CxTR1	Res.	DAP	Res.	DWX	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2[1:0]		Res.	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2[1:0]	
	Reset value		0		0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0		0	0	0	0	0	0	0	0	0	0
0x094 + 0x080 * x (x=0 to 15)	HPDMA_CxTR2	TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	Res.	TRIGSEL[5]	TRIGSEL[4:0]					TRIGM[1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	REQSEL[7]	REQSEL[6]	REQSEL[5]	REQSEL[4:0]				
	Reset value	0	0					0	0			0	0	0	0	0	0	0	0		0	0	0	0		0	0	0	0	0	0	0	0	0
0x098 + 0x080 * x (x=0 to 11)	HPDMA_CxBR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BNDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x098+ 0x080 * x (x=12 to 15)	HPDMA_CxBR1	BRDDEC	BRSDEC	DDEC	SDEC	Res.	BRC[10:0]											BNDT[15:0]															
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x09C + 0x080 * x (x=0 to 15)	HPDMA_CxSAR	SA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0 + 0x080 * x (x=0 to 15)	HPDMA_CxDAR	DA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4 + 0x080 * x (x=12 to 15)	HPDMA_CxTR3	Res.		Res.	DAO[12:0]													Res.		Res.	SAO[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A8 + 0x080 * x (x=12 to 15)	HPDMA_CxBR2	BRDAO[15:0]																BRSAO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0CC + 0x080 * x (x=0 to 11)	HPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0CC + 0x080 * x (x=12 to 15)	HPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

13 General purpose direct memory access controller (GPDMA)

13.1 GPDMA introduction

The general purpose direct memory access (GPDMA) controller is a bus master and system peripheral.

The GPDMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories via linked-lists, upon the control of an off-loaded CPU.

13.2 GPDMA main features

- Dual bidirectional AHB master
- Memory-mapped data transfers from a source to a destination:
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
  - Peripheral-to-peripheral

  Transfer arbitration based on a 4-grade programmed priority at channel level:
  - One high-priority traffic class, for time-sensitive channels (queue 3)
  - Three low-priority traffic classes, with a weighted round-robin allocation for non time-sensitive channels (queues 0, 1, 2)
- Per channel event generation, on any of the following events: transfer complete, half-transfer complete, data transfer error, user setting error, link transfer error, completed suspension, and trigger overrun
- Per channel interrupt generation, with separately programmed interrupt enable per event
- 16 concurrent GPDMA channels:
  - Per channel FIFO for queuing source and destination transfers (see *Section 13.3.1*)
  - Intra-channel GPDMA transfers chaining via programmable linked-list into memory, supporting two execution modes: run-to-completion and link step mode
  - Intra-channel and inter-channel GPDMA transfers chaining via programmable GPDMA input triggers connection to GPDMA task completion events
- Per linked-list item within a channel:
  - Separately programmed source and destination transfers
  - Programmable data handling between source and destination: byte-based reordering, packing or unpacking, padding or truncation, sign extension and left/right realignment
  - Programmable number of data bytes to be transferred from the source, defining the block level and byte-based padding
  - Linear source and destination addressing: either fixed or contiguously incremented addressing, programmed at a block level, between successive burst transfers

  - 2D source and destination addressing: programmable signed address offsets between successive burst transfers (non-contiguous addressing within a block, combined with programmable signed address offsets between successive blocks, at a second 2D/repeated block level, for a reduced set of channels (see *Section 13.3.1*)
  - Support for scatter-gather (multi-buffer transfers), data interleaving and deinterleaving via 2D addressing
  - Programmable GPDMA request and trigger selection
  - Programmable GPDMA half-transfer and transfer complete events generation
  - Pointer to the next linked-list item and its data structure in memory, with automatic update of the GPDMA linked-list control registers
  - Channel abort and restart
  - Channel terminate with FIFO draining
- Debug:
  - Channel suspend and resume support
  - Channel status reporting, including FIFO level, and event flags
- Privileged/unprivileged support:
  - Support for privileged and unprivileged GPDMA transfers, independently at a channel level
  - Privileged-aware AHB slave port

13.3 GPDMA implementation

13.3.1 GPDMA channels

A given GPDMA channel x is implemented with the following features and intended usage. To make the best use of the GPDMA performances, the table below lists some general recommendations, allowing the user to select and allocate a channel, given its implemented FIFO size and the requested GPDMA transfer.

Table 93. GPDMA1 channel implementation

Channel x	Hardware parameters		Features
	dma_fifo_size[x]	dma_addressing[x]	
x = 0 to 11	2	0	Channel x (x = 0 to 11) is implemented with: – a FIFO of 8 bytes, 2 words – fixed/contiguously incremented addressing These channels must be typically allocated for GPDMA transfers between an APB or AHB peripheral and AHB SRAM.
x = 12 to 15	4	1	Channel x (x = 12 to 15) is implemented with: – a FIFO of 32 bytes, 8 words – 2D addressing These channels may be also used for GPDMA transfers, between a demanding AHB peripheral and AHB SRAM, or for transfers from/to AXI SRAM or external AXI memories.

13.3.2 GPDMA in low-power modes

The GPDMA wake-up feature is implemented in the device low-power modes as per the table below.

Table 94. GPDMA1 wake-up in low-power modes

Feature	Low-power modes
Wake-up	GPDMA1 in Sleep mode

13.3.3 GPDMA requests

A GPDMA request from a peripheral can be assigned to a GPDMA channel x, via REQSEL[7:0] in GPDMA_CxTR2, provided that SWREQ = 0.

The GPDMA requests mapping is specified in the table below.

Table 95. Programmed GPDMA1 request

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
0	adc1_dma
1	adc2_dma
2	adc3_dma
3	adc4_dma
4	dac1_ch1_dma
5	dac1_ch2_dma
6	dac2_ch1_dma
7	dac2_ch2_dma
8	dac3_ch1_dma
9	dac3_ch2_dma
10	dac4_ch1_dma
11	dac4_ch2_dma
12	hash_dma
13	hrtim1_master_dma
14	hrtim1_tima_dma
15	hrtim1_timb_dma
16	hrtim1_timc_dma
17	hrtim1_timd_dma
18	hrtim1_time_dma
19	hrtim1_timf_dma
20	hrtim2_master_dma
21	hrtim2_tima_dma
22	hrtim2_timb_dma

Table 95. Programmed GPDMA1 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
23	hrtim2_timc_dma
24	hrtim2_timd_dma
25	hrtim2_time_dma
26	hrtim2_timf_dma
27	tim1_ch1_dma
28	tim1_ch2_dma
29	tim1_ch3_dma
30	tim1_ch4_dma
31	tim1_upd_dma
32	tim1_trgi_dma
33	tim1_com_dma
34	tim8_ch1_dma
35	tim8_ch2_dma
36	tim8_ch3_dma
37	tim8_ch4_dma
38	tim8_upd_dma
39	tim8_trgi_dma
40	tim8_com_dma
41	tim20_ch1_dma
42	tim20_ch2_dma
43	tim20_ch3_dma
44	tim20_ch4_dma
45	tim20_upd_dma
46	tim20_trgi_dma
47	tim20_com_dma
48	tim2_ch1_dma
49	tim2_ch2_dma
50	tim2_ch3_dma
51	tim2_ch4_dma
52	tim2_upd_dma
53	tim2_trgi_dma
54	tim3_ch1_dma
55	tim3_ch2_dma
56	tim3_ch3_dma
57	tim3_ch4_dma

Table 95. Programmed GPDMA1 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
58	tim3_upd_dma
59	tim3_trgi_dma
60	tim4_ch1_dma
61	tim4_ch2_dma
62	tim4_ch3_dma
63	tim4_ch4_dma
64	tim4_upd_dma
65	tim4_trgi_dma
66	tim5_ch1_dma
67	tim5_ch2_dma
68	tim5_ch3_dma
69	tim5_ch4_dma
70	tim5_upd_dma
71	tim5_trgi_dma
72	tim6_upd_dma
73	tim7_upd_dma
74	tim15_ch1_dma
75	tim15_ch2_dma
76	tim15_upd_dma
77	tim15_trgi_dma
78	tim15_com_dma
79	tim16_ch1_dma
80	tim16_upd_dma
81	tim16_com_dma
82	tim17_ch1_dma
83	tim17_upd_dma
84	tim17_com_dma
85	spi1_rx_dma
86	spi1_tx_dma
87	spi2_rx_dma
88	spi2_tx_dma
89	spi3_rx_dma
90	spi3_tx_dma
91	spi4_rx_dma
92	spi4_tx_dma

Table 95. Programmed GPDMA1 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
93	spi5_rx_dma
94	spi5_tx_dma
95	spi6_rx_dma
96	spi6_tx_dma
97	i2c1_rx_dma
98	i2c1_tx_dma
99	i2c2_rx_dma
100	i2c2_tx_dma
101	i2c3_rx_dma
102	i2c3_tx_dma
103	i3c1_rx_dma
104	i3c1_tx_dma
105	i3c1_tc_dma
106	i3c1_rs_dma
107	usart1_rx_dma
108	usart1_tx_dma
109	usart2_rx_dma
110	usart2_tx_dma
111	usart3_rx_dma
112	usart3_tx_dma
113	uart4_rx_dma
114	uart4_tx_dma
115	uart5_rx_dma
116	uart5_tx_dma
117	usart6_rx_dma
118	usart6_tx_dma
119	uart7_rx_dma
120	uart7_tx_dma
121	uart8_rx_dma
122	uart8_tx_dma
123	uart9_rx_dma
124	uart9_tx_dma
125	usart10_rx_dma
126	usart10_tx_dma
127	cordic_rd_dma

Table 95. Programmed GPDMA1 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
128	cordic_wr_dma
129	fmac1_rd_dma
130	fmac1_wr_dma
131	lptim1_ic1_dma
132	lptim1_ic2_dma
133	lptim1_ue_dma
134	lptim2_ic1_dma
135	lptim2_ic2_dma
136	lptim2_ue_dma
137	lptim3_ic1_dma
138	lptim3_ic2_dma
139	lptim3_ue_dma
140	adf1_flt0_dma
141	usbpd_tx_dma
142	usbpd_rx_dma
143	lpuart1_rx_dma
144	lpuart1_tx_dma
145	-

13.3.4 GPDMA block requests

Some GPDMA requests must be programmed as a block request, and not as a burst request. Then BREQ in GPDMA_CxTR2 must be set for a correct GPDMA execution of the requested peripheral transfer at the hardware level.

Table 96. Programmed GPDMA1 request as a block request

GPDMA block requests
lptim1_ue_dma
lptim2_ue_dma
lptim3_ue_dma

13.3.5 GPDMA channels with peripheral early termination

A GPDMA channel, if implemented with this feature, can support the early termination of the data transfer from the peripheral which does also support this feature.

Table 97. GPDMA1 channel with peripheral early termination

GPDMA channel x with peripheral early termination
x = 0 and x = 15

This GPDMA support is activated when the channel x is programmed with GPDMA_CxTR2.PFREQ = 1. Then, the peripheral itself can initiate and request a data transfer completion, before the GPDMA has transferred the whole block (see *Section 13.4.15* for more details).

Table 98. Programmed GPDMA1 request with peripheral early termination

Programmed GPDMA channel x request with peripheral early termination
i3c1_rx_dma

13.3.6 GPDMA triggers

A GPDMA trigger can be assigned to a GPDMA channel x, via TRIGSEL[5:0] in GPDMA_CxTR2, provided that TRIGPOL[1:0] defines a rising or a falling edge of the selected trigger (TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

Table 99. Programmed GPDMA1 trigger

GPDMA_CxTR2.TRIGSEL[5:0]	Selected GPDMA trigger
0	tim1_trgo
1	tim1_trgo2
2	tim2_trgo
3	tim3_trgo
4	tim4_trgo
5	tim5_trgo
6	tim6_trgo
7	tim7_trgo
8	tim8_trgo
9	tim8_trgo2
10	tim9_trgo
11	tim12_trgo
12	tim15_trgo
13	tim20_trgo
14	tim20_trgo2
15	hsp1_trgo0

Table 99. Programmed GPDMA1 trigger (continued)

GPDMA_CxTR2.TRIGSEL[5:0]	Selected GPDMA trigger
16	hsp1_trgo1
17	hsp1_trgo2
18	hsp1_trgo3
19	lptim1_ch1
20	lptim1_ch2
21	lptim2_ch1
22	lptim2_ch2
23	lptim3_ch1
24	lptim3_ch2
25	lptim4_out
26	lptim5_out
27	exti_exti0_wkup
28	exti_wkup_rtc_wkup
29	lpuart1_it_r_wup_async
30	lpuart1_it_t_wup_async
31	spi6_it
32	gpdma1_ch0_tc
33	gpdma1_ch1_tc
34	gpdma1_ch2_tc
35	gpdma1_ch3_tc
36	gpdma1_ch4_tc
37	gpdma1_ch5_tc
38	gpdma1_ch6_tc
39	gpdma1_ch7_tc
40	gpdma1_ch8_tc
41	gpdma1_ch9_tc
42	gpdma1_ch10_tc
43	gpdma1_ch11_tc
44	gpdma1_ch12_tc
45	gpdma1_ch13_tc
46	gpdma1_ch14_tc
47	gpdma1_ch15_tc
48	hpdma1_ch0_tc
49	hpdma1_ch1_tc
50	hpdma1_ch2_tc

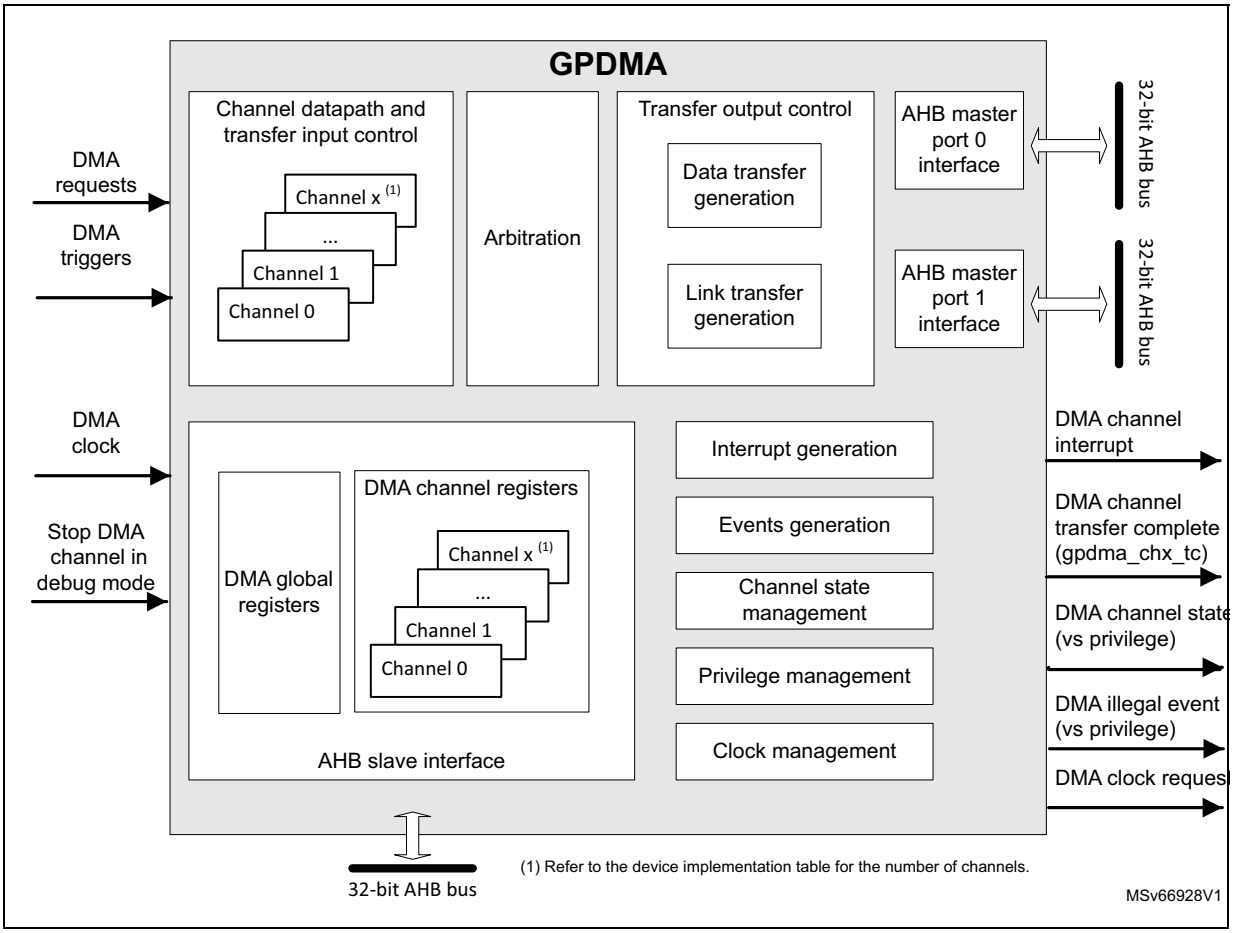
Table 99. Programmed GPDMA1 trigger (continued)

GPDMA_CxTR2.TRIGSEL[5:0]	Selected GPDMA trigger
51	hpdma1_ch3_tc
52	hpdma1_ch4_tc
53	hpdma1_ch5_tc
54	hpdma1_ch6_tc
55	hpdma1_ch7_tc
56	hpdma1_ch8_tc
57	hpdma1_ch9_tc
58	hpdma1_ch10_tc
59	hpdma1_ch11_tc
60	hpdma1_ch12_tc
61	hpdma1_ch13_tc
62	hpdma1_ch14_tc
63	hpdma1_ch15_tc

13.4 GPDMA functional description

13.4.1 GPDMA block diagram

Figure 101. GPDMA block diagram



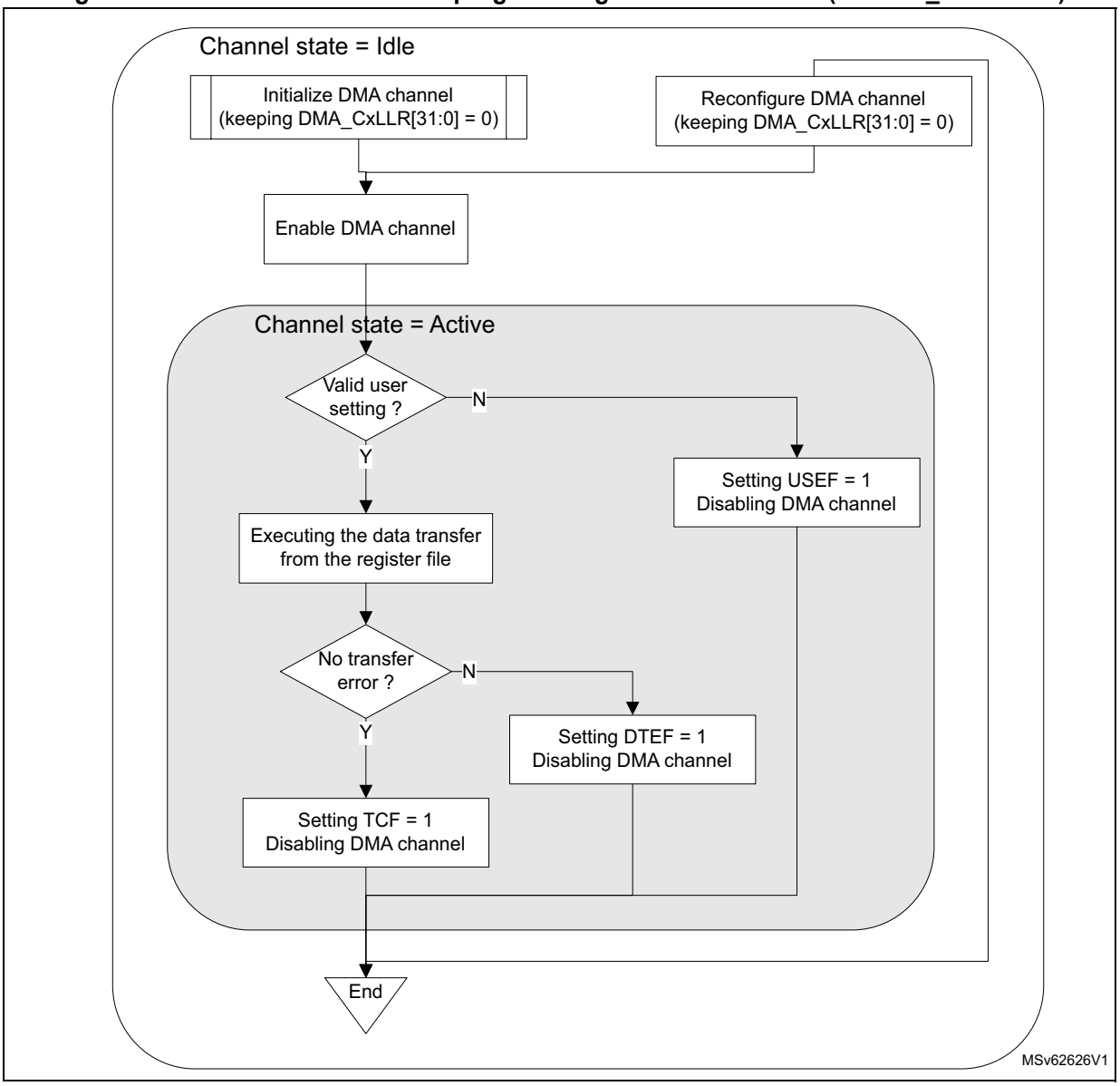
13.4.2 GPDMA channel state and direct programming without any linked-list

After a GPDMA reset, a GPDMA channel x is in idle state. When the software writes 1 in GPDMA_CxCR.EN, the channel takes into account the value of the different channel configuration registers (GPDMA_CxXXX), switches to the active/non-idle state and starts to execute the corresponding requested data transfers.

After enabling/starting a GPDMA channel transfer by writing 1 in GPDMA_CxCR.EN, a GPDMA channel interrupt on a complete transfer notifies the software that the GPDMA channel is back in idle state (EN is then deasserted by hardware) and that the channel is ready to be reconfigured then enabled again.

Figure 102 illustrates this GPDMA direct programming without any linked-list (GPDMA_CxLLR = 0).

Figure 102. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0)



13.4.3 GPDMA channel suspend and resume

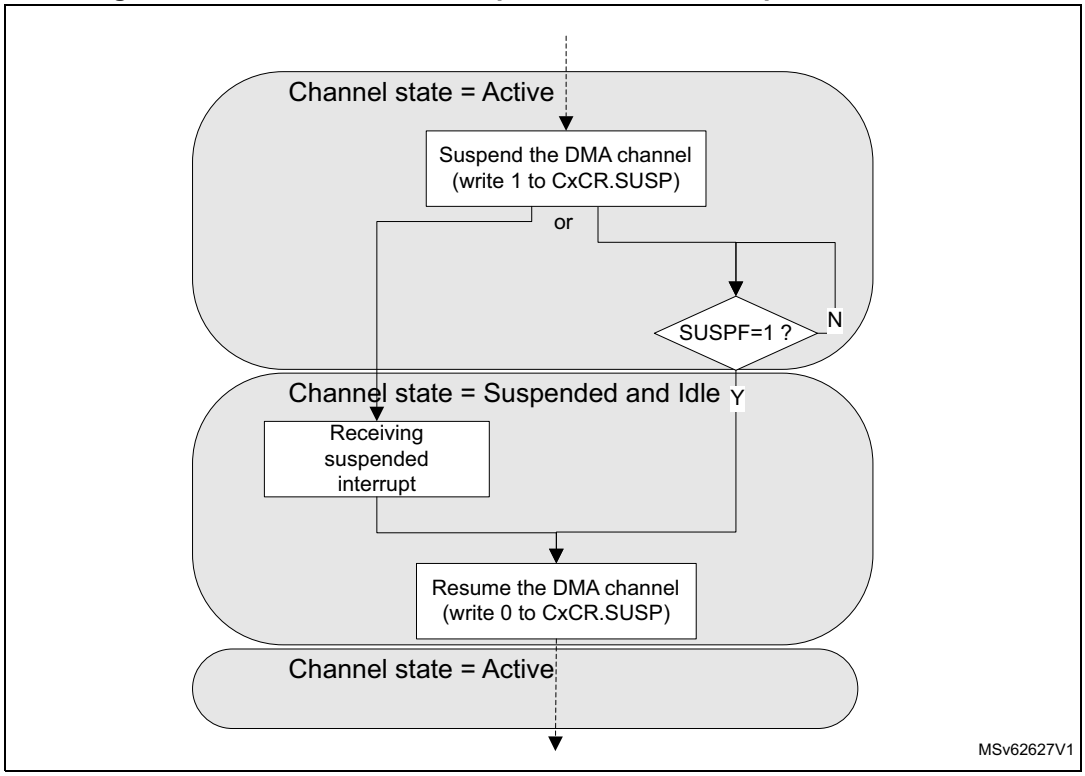
The software can suspend on its own a channel still active, with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls the suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master ports. Then the software can observe, in a steady state, any read register or register field that is hardware modifiable.

Note: An ongoing GPDMA transfer can be a data transfer (a source/destination burst transfer) or a link transfer for the internal update of the linked-list register file from the next linked-list item.

3. The software safely resumes the suspended channel by writing 0 to GPDMA_CxCR.SUSP.

Figure 103. GPDMA channel suspend and resume sequence



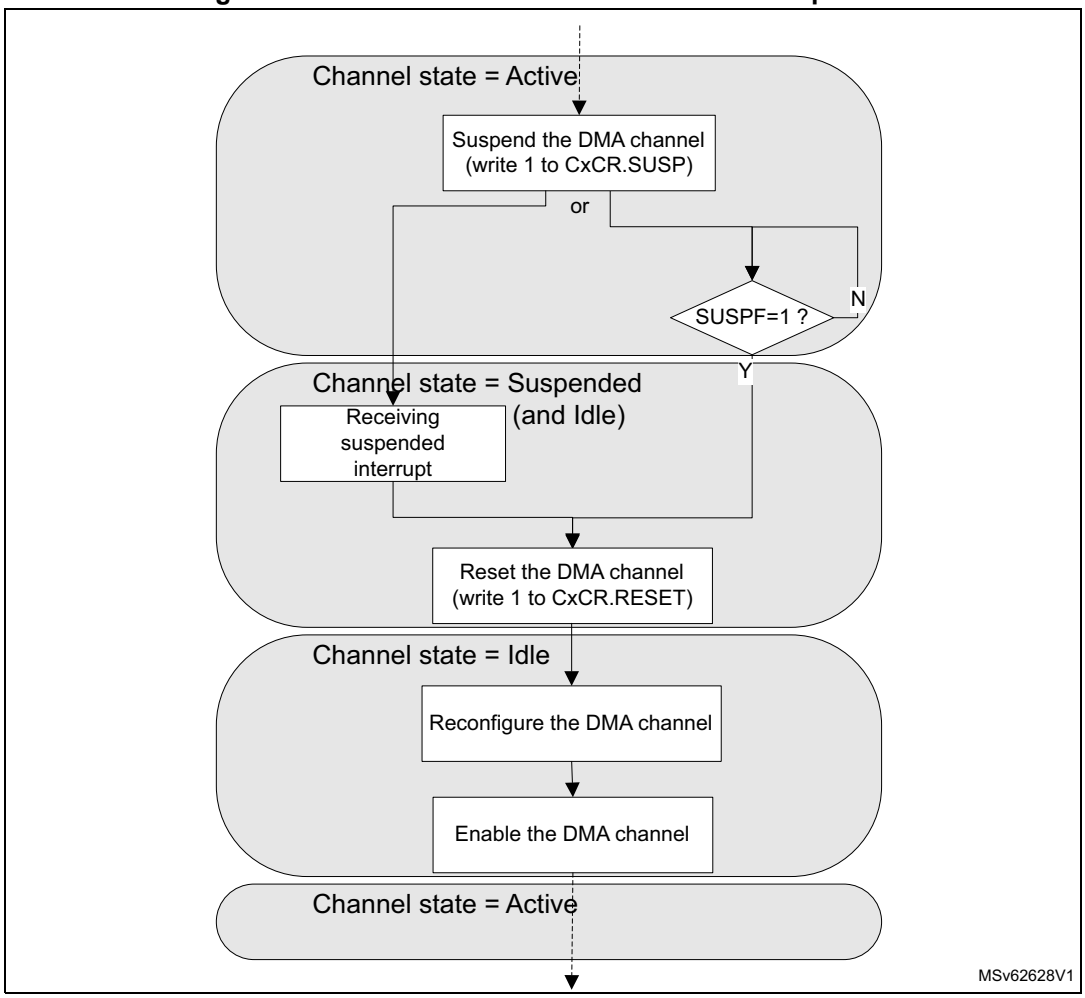
Note: A suspend and resume sequence does not impact the GPDMA_CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.

13.4.4 GPDMA channel abort and restart

Alternatively, like for aborting a continuous GPDMA transfer with a circular buffering or a double buffering, the software can abort, on its own, a still active channel with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master port.
3. The software resets the channel by writing 1 to GPDMA_CxCR.RESET. This causes the reset of the FIFO, the reset of the channel internal state, the reset of the GPDMA_CxCR.EN bit, and the reset of the GPDMA_CxCR.SUSP bit.
4. The software safely reconfigures the channel. The software must reprogram the hardware-modified GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR registers.
5. In order to restart the aborted then reprogrammed channel, the software enables it again by writing 1 to the GPDMA_CxCR.EN bit.

Figure 104. GPDMA channel abort and restart sequence

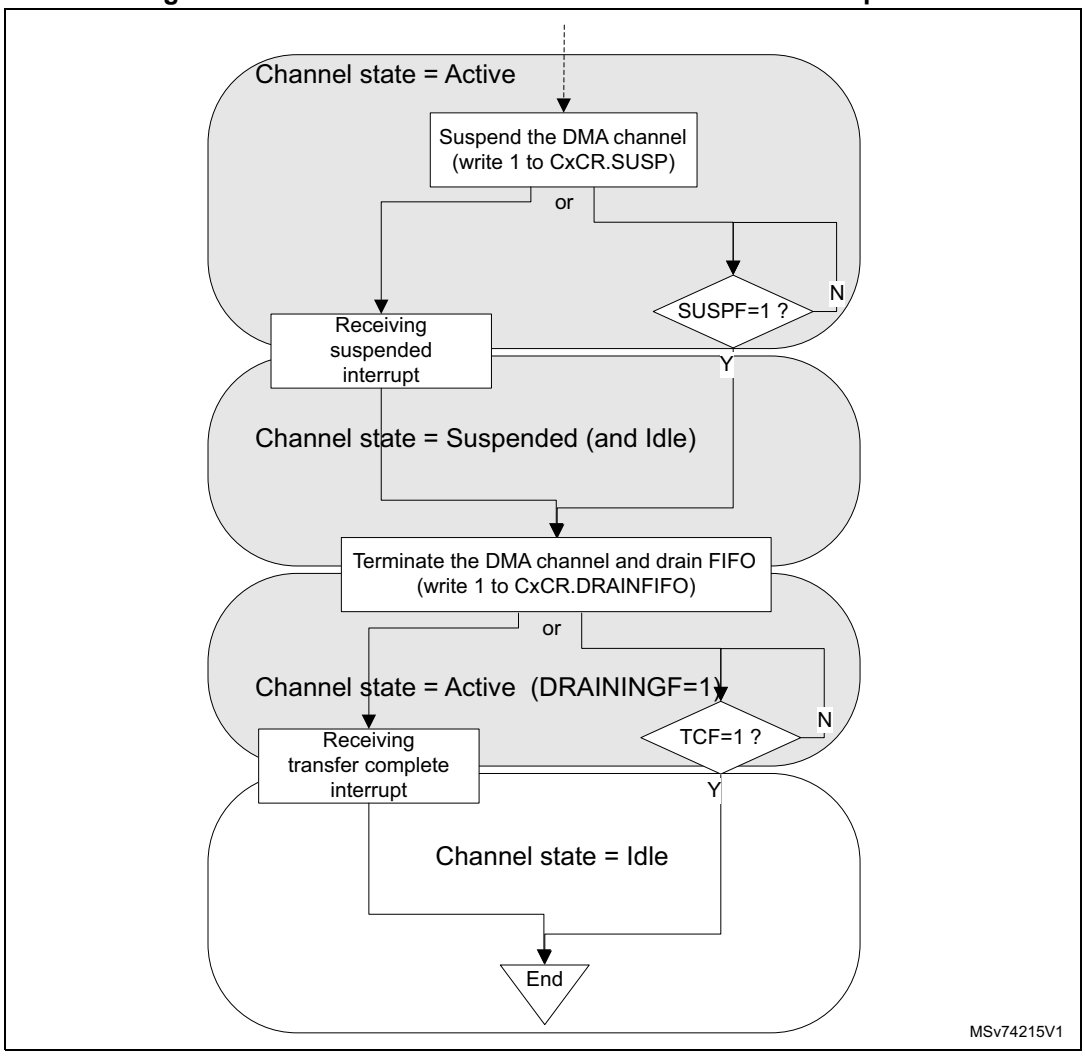


13.4.5 GPDMA channel terminate and drain FIFO

Alternatively, for terminating a continuous GPDMA transfer with a circular buffering or a double buffering, and having popped out any residual byte(s) in the channel FIFO and writing them to the destination, the software can terminate an active channel and drain the channel FIFO with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master port.
3. The software orders to drain the channel FIFO by writing 1 to GPDMA_CxCR.DRAINFIFO.
4. The software polls transfer complete flag GPDMA_CxSR.TCF until TCF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.TCIE. Wait for the channel to be effectively with a drained FIFO means wait for completion of any residual write burst from the FIFO to the destination. Then the channel is disabled by hardware and in IDLE state.

Figure 105. GPDMA channel terminate and drain FIFO sequence



Note: If the softwares writes the GPDMA_CxCR register in a single shot with concurrently SUSP = 1, DRAINFIFO = 1, and RESET = 0, then the sequence for terminating and draining the FIFO also occurs.

GPDMA_CxTR2.DREQ = 1 and GPDMA_CxCR.DRAINFIFO = 1 is not supported, only peripheral-to-memory and memory-to-memory transfers are supported to be terminated with drained FIFO.

13.4.6 GPDMA linked-list data structure

Alternatively to the direct programming mode, a channel can be programmed by a list of transfers, known as a list of linked-list items (LLI). Each LLI is defined by its data structure.

The base address in memory of the data structure of a next LLI_{n+1} of a channel x is the sum of the following:

- the link base address of the channel x (in GPDMA_CxLBAR)
- the link address offset (LA[15:2] field in GPDMA_CxLLR). The linked-list register GPDMA_CxLLR is the updated result from the data structure of the previous LLIn of the channel x.

The data structure for each LLI may be specific.

A linked-list data structure is addressed following the value of the following UT1, UT2, UB1, USA, UDA, UT3 and UB2 if present, and ULL bits in GPDMA_CxLLR.

In linked-list mode, each GPDMA linked-list register (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, GPDMA_CxTR3 and GPDMA_CxBR2 if present, and GPDMA_CxLLR) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of the GPDMA_CxLLR register that was conditionally updated from the linked-list data structure of the previous LLI.

Caution: The user must program the pointer to the next linked-list data structure (GPDMA_CxLLR[15:0]) not to exceed the 64-Kbyte addressable space defined by the link base address register (GPDMA_CxLBAR). The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Static linked-list data structure

For example, when the update bits (UT1, UT2, UB1, USA, UDA, plus UB2 and UT3 if present, and ULL) in GPDMA_CxLLR are all asserted, the linked-list data structure in the memory is maximal with:

- channel x (x = 0 to 11) contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR (see *Figure 106*) and including the first linked-list register file (LLI_0) and the next LLIs (such as LLI_1, LLI_2) in the memory
- channel x (x = 12 to 15), contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, GPDMA_CxTR3, GPDMA_CxBR2 and GPDMA_CxLLR, (see *Figure 107*), and including the first linked-list register file (LLI_0) and the next LLIs (such as LLI_1, LLI_2) in the memory

Figure 106. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x

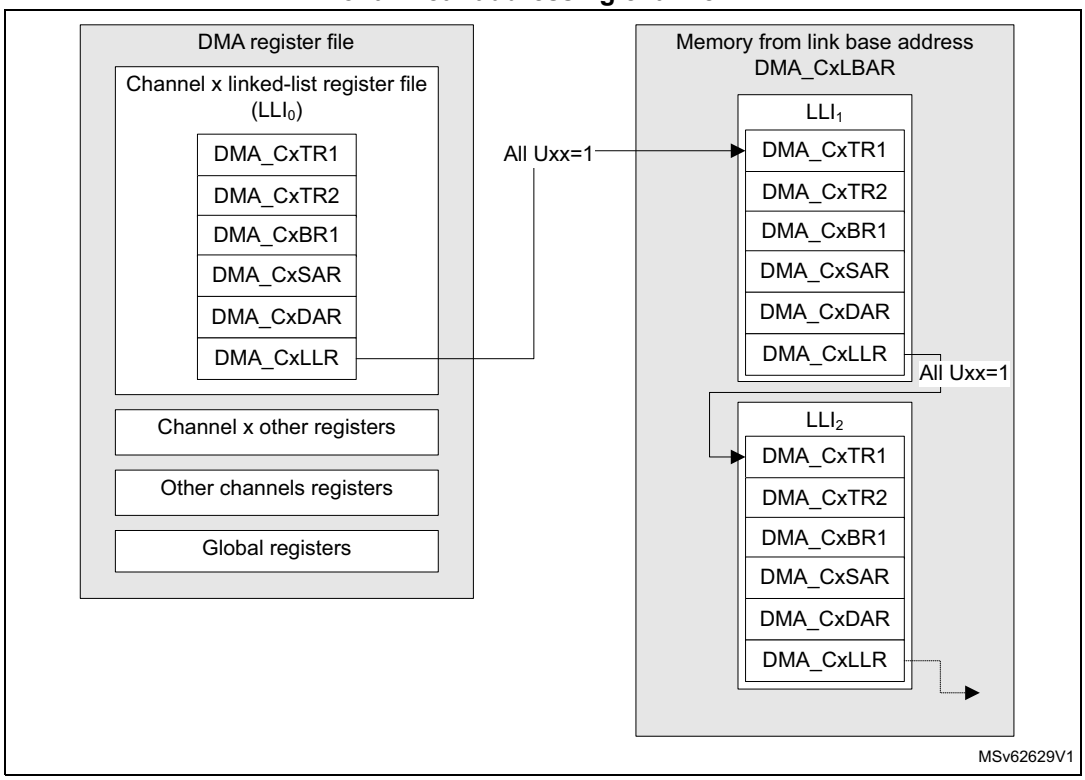
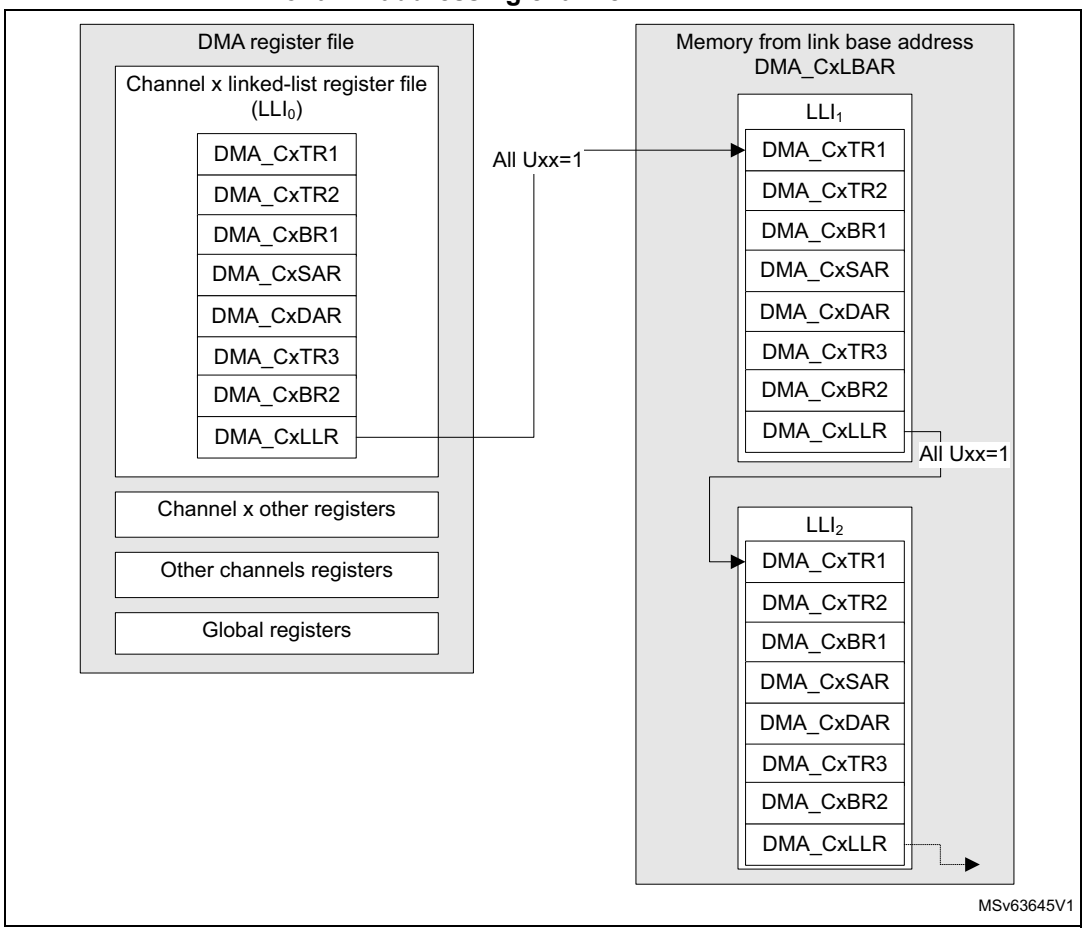


Figure 107. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x



Dynamic linked-list data structure

Alternatively, the memory organization for the full list of LLIs can be compacted with specific data structure for each LLI.

If UT1 = 0 and UT2 = 1, the link address offset of the register GPDMA_CxLLR is pointing to the updated value of the GPDMA_CxTR2 instead of the GPDMA_CxTR1 which is not to be modified (see *Figure 108*).

Example: if UT1 = UB1 = USA = 0 and if UT3 = UB2 = 0, when channel x is with 2D addressing, and if UT2 = UDA = ULL = 1, the next LLI does not contain an (updated) value for GPDMA_CxTR1, nor GPDMA_CxBR1, nor GPDMA_CxSAR, nor GPDMA_CxTR3, nor GPDMA_CxBR2 when channel x is with 2D addressing. The next LLI contains an updated value for GPDMA_CxTR2, GPDMA_CxDAR, and GPDMA_CxLLR, as shown in *Figure 109*.

Figure 108. GPDMA dynamic linked-list data structure of a linear addressing channel x

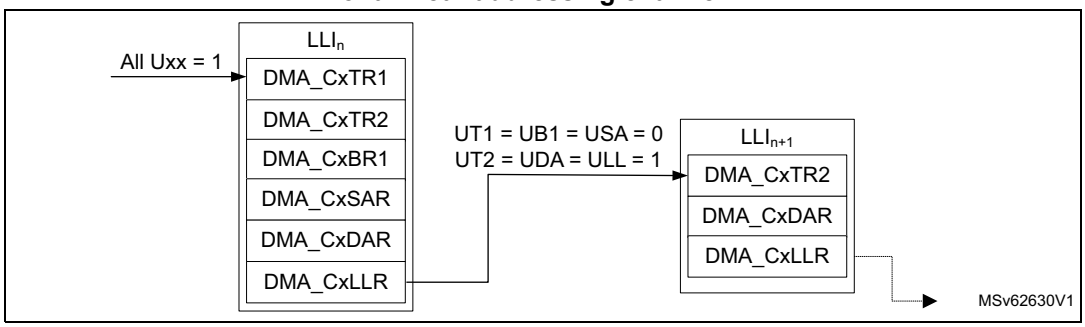
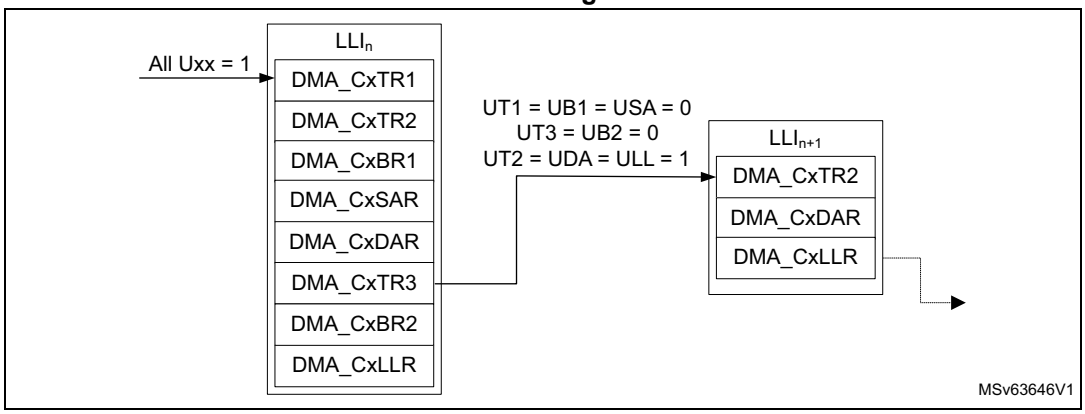


Figure 109. GPDMA dynamic linked-list data structure of a 2D addressing channel x



13.4.7 Linked-list item transfer execution

A LLI_n transfer is the sequence of:

1. a data transfer: GPDMA executes the data transfer as described by the GPDMA internal register file (this data transfer can be void/null for LLI_0)
2. a conditional link transfer: GPDMA automatically and conditionally updates its internal register file by the data structure of the next LLI_{n+1}, as defined by the GPDMA_CxLLR value of the LLI_n.

Note: The initial data transfer as defined by the internal register file (LLI_0) can be null (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxTR2.PFREQ = 0) provided that the conditional update bit UB1 in GPDMA_CxLLR is set (meaning there is a non-null data transfer described by the next LLI_1 in the memory to be executed).

Depending on the intended GPDMA usage, a GPDMA channel x can be executed as described by the full linked-list (run-to-completion mode, GPDMA_CxCR.LSM = 0) or a GPDMA channel x can be programmed for a single execution of a LLI (link step mode, GPDMA_CxCR.LSM = 1), as described in the next sections.

13.4.8 GPDMA channel state and linked-list programming in run-to-completion mode

When GPDMA_CxCR.LSM = 0 (in full list execution mode, execution of the full sequence of LLIs, named run-to-completion mode), a GPDMA channel x is initially programmed, started

by writing 1 to GPDMA_CxCR.EN, and after completed at channel level. The channel transfer is:

- configured with at least the following:
  - the first LLI_0, internal linked-list register file: GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2
  - the last LLI_N, described by the linked-list data structure in memory, as defined by the GPDMA_CxLLR reflecting the before last LLI_{N-1}
- completed when GPDMA_CxLLR[31:0] = 0, GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0, at the end of the last LLI_{N-1} transfer

GPDMA_CxLLR[31:0] = 0 is the condition of a linked-list based channel completion and means the following:

- The 16 low significant bits GPDMA_CxLLR.LA[15:0] of the next link address are null.
- All the update bits GPDMA_CxLLR.Uxx are null (UT1, UT2, UB1, USA, UDA and ULL, plus UB2 and UT3).

The channel may never be completed when GPDMA_CxLLR.LSM = 0:

- If the last LLI_N is recursive, pointing to itself as a next LLI:
  - either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated with the same value
  - or GPDMA_CxLLR.ULL = 0
- If LLI_N is pointing to a previous LLI

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0 (if present). Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 13.4.15* for more details).

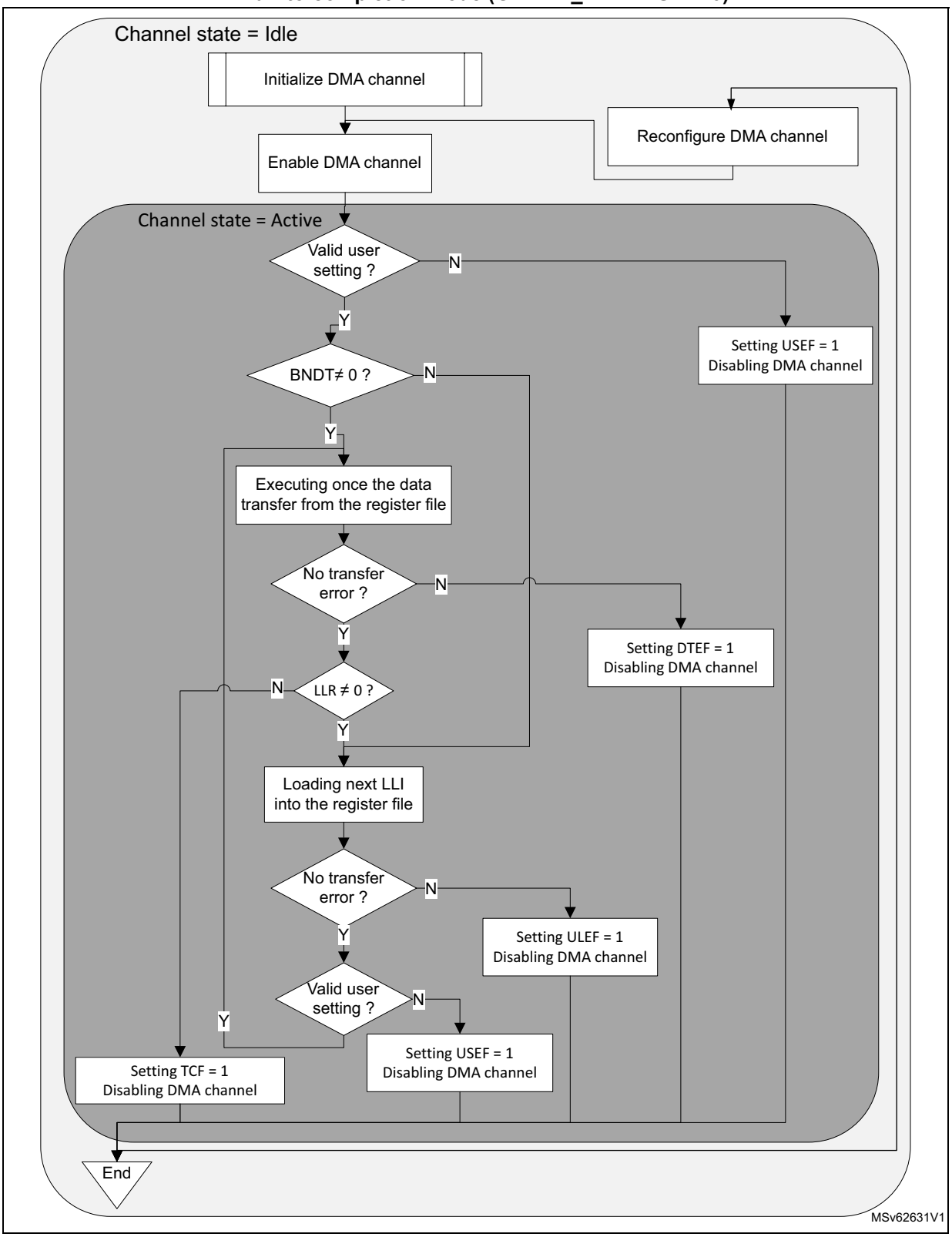
In typical run-to-completion mode, allocation of an GPDMA channel, including detailed programming, occurs once during GPDMA initialization. This process reserves a data communication link and GPDMA service during run-time for repeated transfers, such as transfers from a peripheral to memory, from memory to a peripheral, or memory-to-memory transfers. The reserved data communication link may consist of a dedicated channel, or the channel may be shared. A repeated transfer consists of a sequence of linked list items (LLIs).

Figure 110 depicts the GPDMA channel execution and its registers programming in run-to-completion mode.

Note: Figure 110 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at channel completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion, the hardware always set TCEF = 1 and disables the channel.

In Figure 110, BNDT ≠ 0 is the typical condition for starting the first data transfer in this figure. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 13.3.5).

Figure 110. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0)



Run-time inserting a LLI_n via an auxiliary channel, in run-to-completion mode

The start of the link transfer of the LLI_{n-1} (start of the LLI_n loading) can be conditioned by the occurrence of a trigger, when programming the following fields of the GPDMA_CxTR2 in the data structure of the LLI_{n-1}:

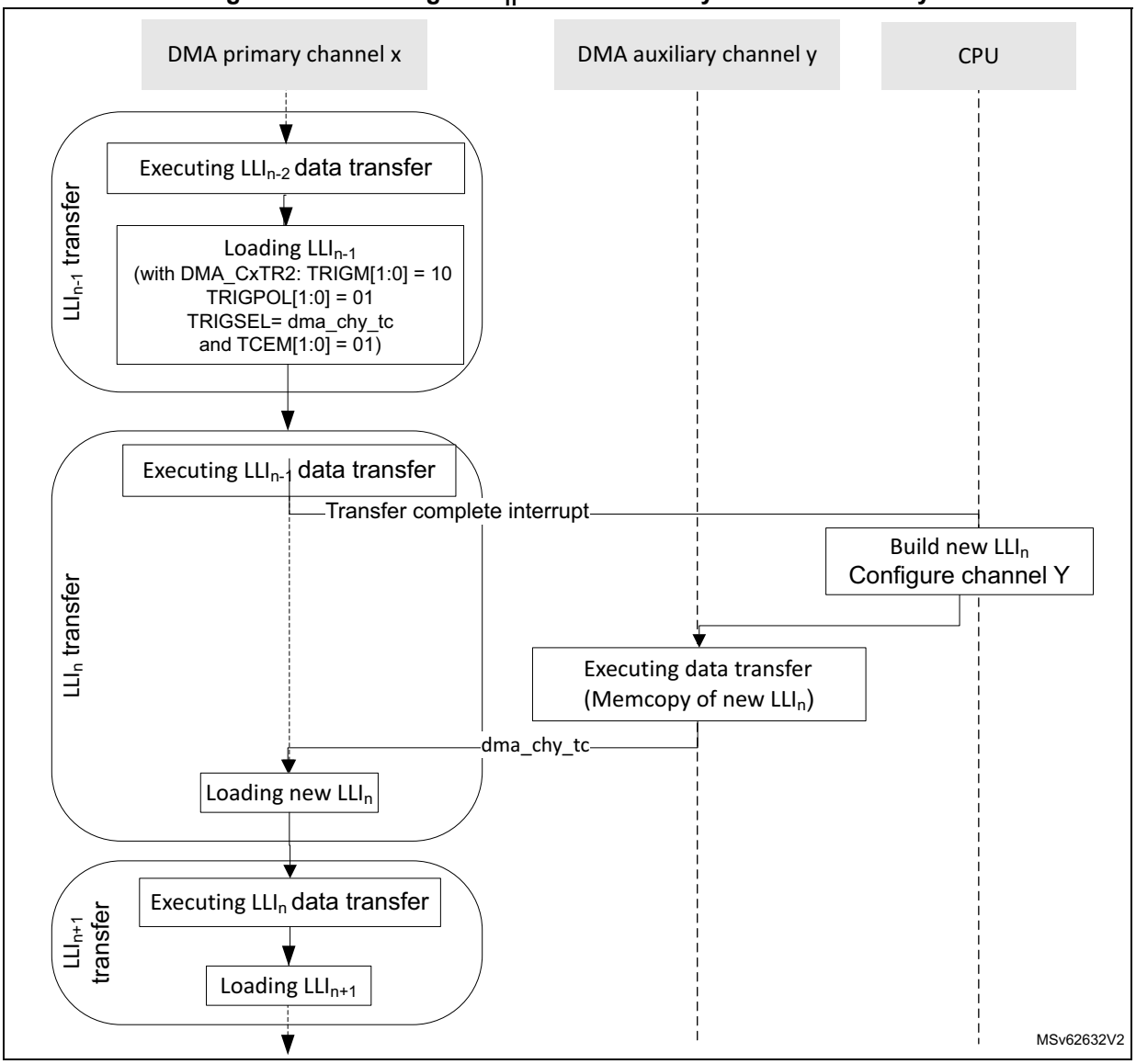
- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[5:0] (see *Section 13.3.6* for the trigger selection details)

Another auxiliary channel y can be used to store the channel x LLI_n in the memory and to generate a transfer complete event gpdma_chy_tc. By selecting this event as the input trigger of the link transfer of the LLI_{n-1} of the channel x, the software can pause the primary channel x after its LLI_{n-1} data transfer, until it is indeed written the LLI_n.

Figure 111 depicts such a dynamic elaboration of a linked-list of a primary channel x, via another auxiliary channel y.

Caution: This use case is restricted to an application with a LLI_{n-1} data transfer that does not need a trigger. The triggering mode of this LLI_{n-1} is used to load the next LLI_n.

Figure 111. Inserting a LLI_n with an auxiliary GPDMA channel y



13.4.9 GPDMA channel state and linked-list programming in link step mode

When GPDMA_CxCR.LSM = 1 (in link step execution mode, single execution of one LLI), a channel transfer is executed and completed after each single execution of a LLI, including its (conditional) data transfer and its (conditional) link transfer.

A GPDMA channel transfer can be programmed at LLI level, started by writing 1 into GPDMA_CxCR.EN, and after completed at LLI level:

- The current LLI_n transfer is described with:
  - GPDMA_CxTR1 defines the source/destination elementary single/burst transfers.
  - GPDMA_CxBR1 defines the number of bytes at a block level (BNDT[15:0]) and, for channel x (x = 12 to 15), the number of blocks at a 2D/repeated block level (BRC[10:0]+1) and the incrementing/decrementing mode for address offsets.

  - GPDMA_CxTR2 defines the input control (request, trigger) and the output control (transfer complete event) of the transfer.
  - GPDMA_CxSAR/GPDMA_CxDAR define the source/destination transfer start address.
  - GPDMA_CxTR3 for channel x (x = 12 to 15) defines the source/destination additional address offset between burst transfers.
  - GPDMA_CxBR2 for channel x (x = 12 to 15) defines the source/destination additional address offset between blocks at a 2D/repeated block level.
  - GPDMA_CxLLR defines the data structure and the address offset of the next LLI_{n+1} in the memory.
- The current LLI_n transfer is completed after the single execution of the current LLI_n:
  - after the (conditional) data transfer completion (when GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0
  - after the (conditional) update of the GPDMA link register file from the data structure of the next LLI_{n+1} in memory

Note: If a LLI is recursive (pointing to itself as a next LLI, either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated with the same value, or GPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 13.4.15* for more details).

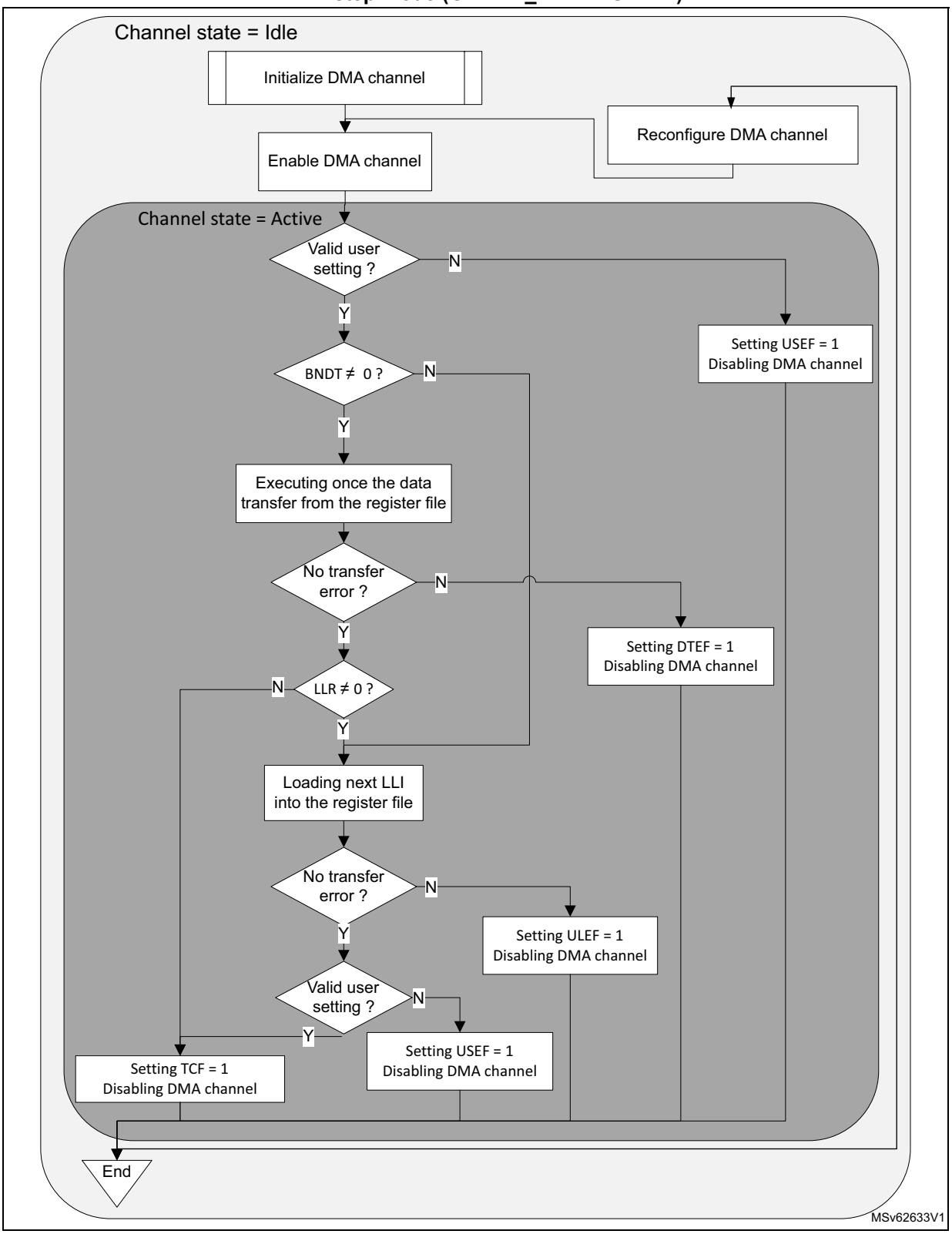
The link step mode can be used to elaborate dynamically LLIs in memory during run-time. The software can be facilitated by using a static data structure for any LLI_n (all update bits of GPDMA_CxLLR have a static value, LLI_n.LLR.LA = LLI_{n-1}.LLR.LA + constant).

Figure 112 depicts the GPDMA channel execution mode, and its programming in link step mode.

Note: Figure 112 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 112, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 13.3.5).

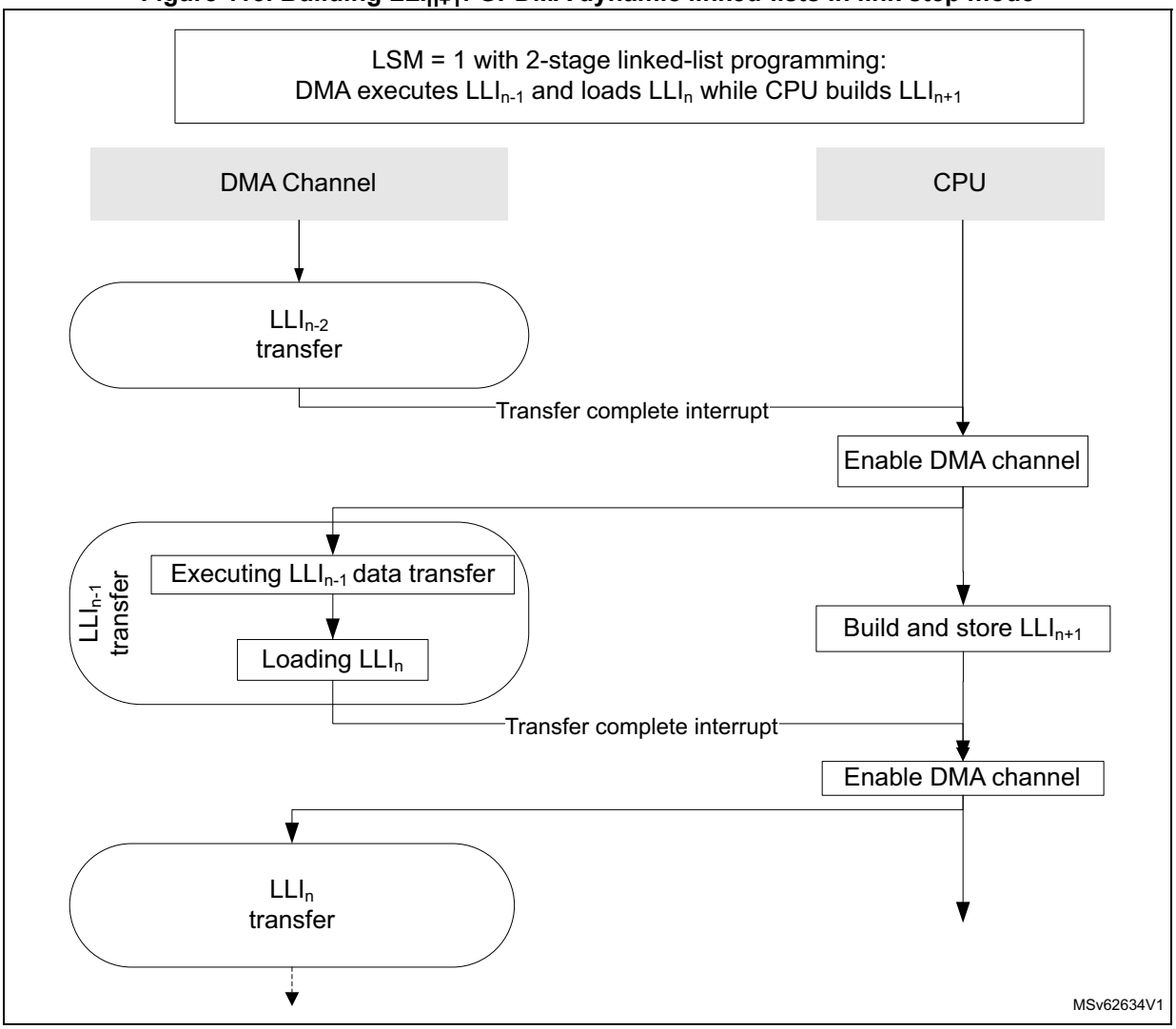
Figure 112. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1)



Run-time adding a LLI_{n+1} in link step mode

During run-time, the software can defer the elaboration of the LLI_{n+1} (and next LLIs), until/after GPDMA executed the transfer from the LLI_{n-1} and loaded the LLI_n from the memory, as shown in *Figure 113*.

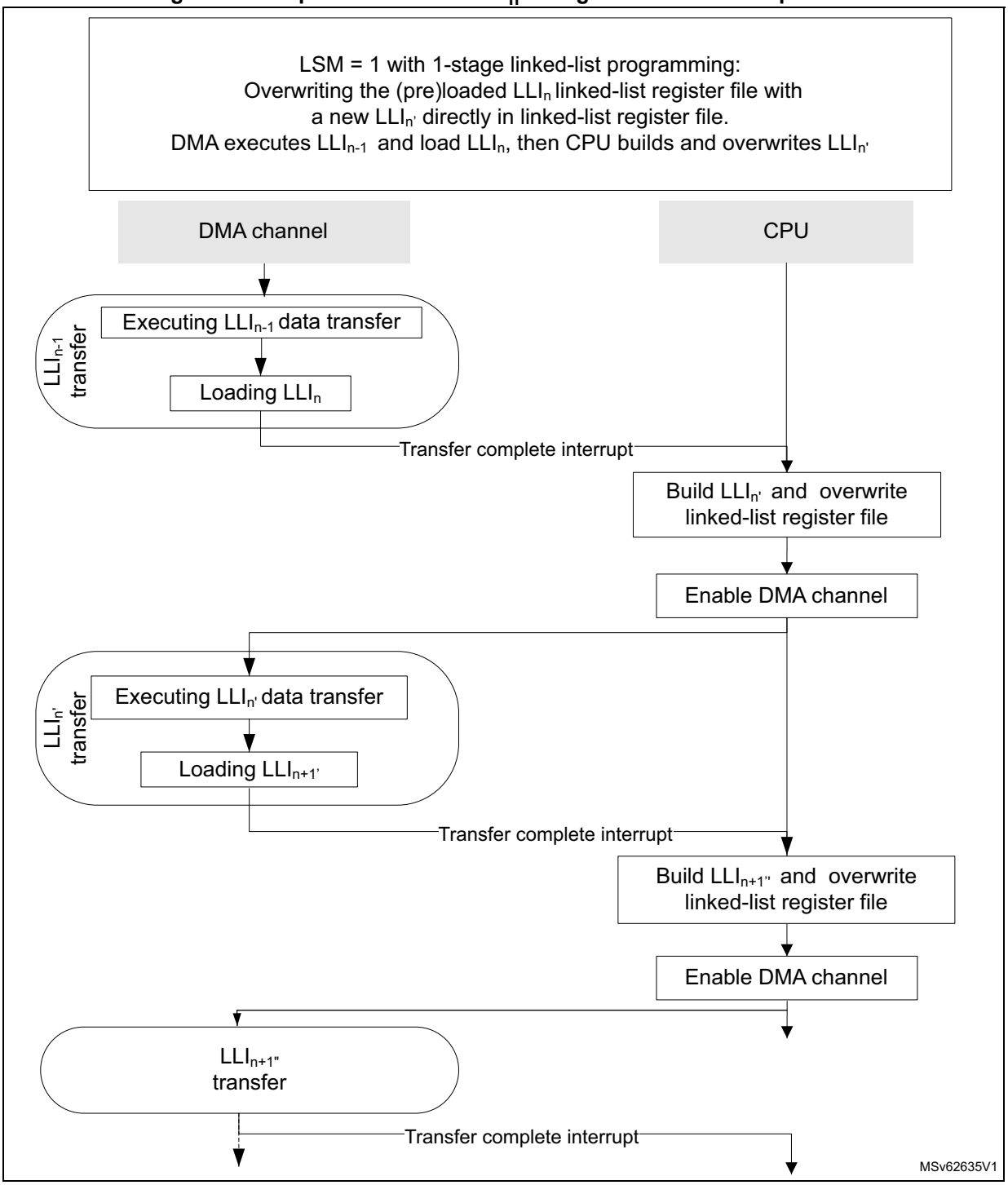
Figure 113. Building LLI_{n+1}: GPDMA dynamic linked-lists in link step mode



Run-time replacing a LLI_n with a new LLI_n', in link step mode (in linked-list register file)

In this link step mode, during run-time, the software can build and insert a new LLI_n', after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory by overwriting directly the linked-list register file with the new LLI_n', as shown in *Figure 114*.

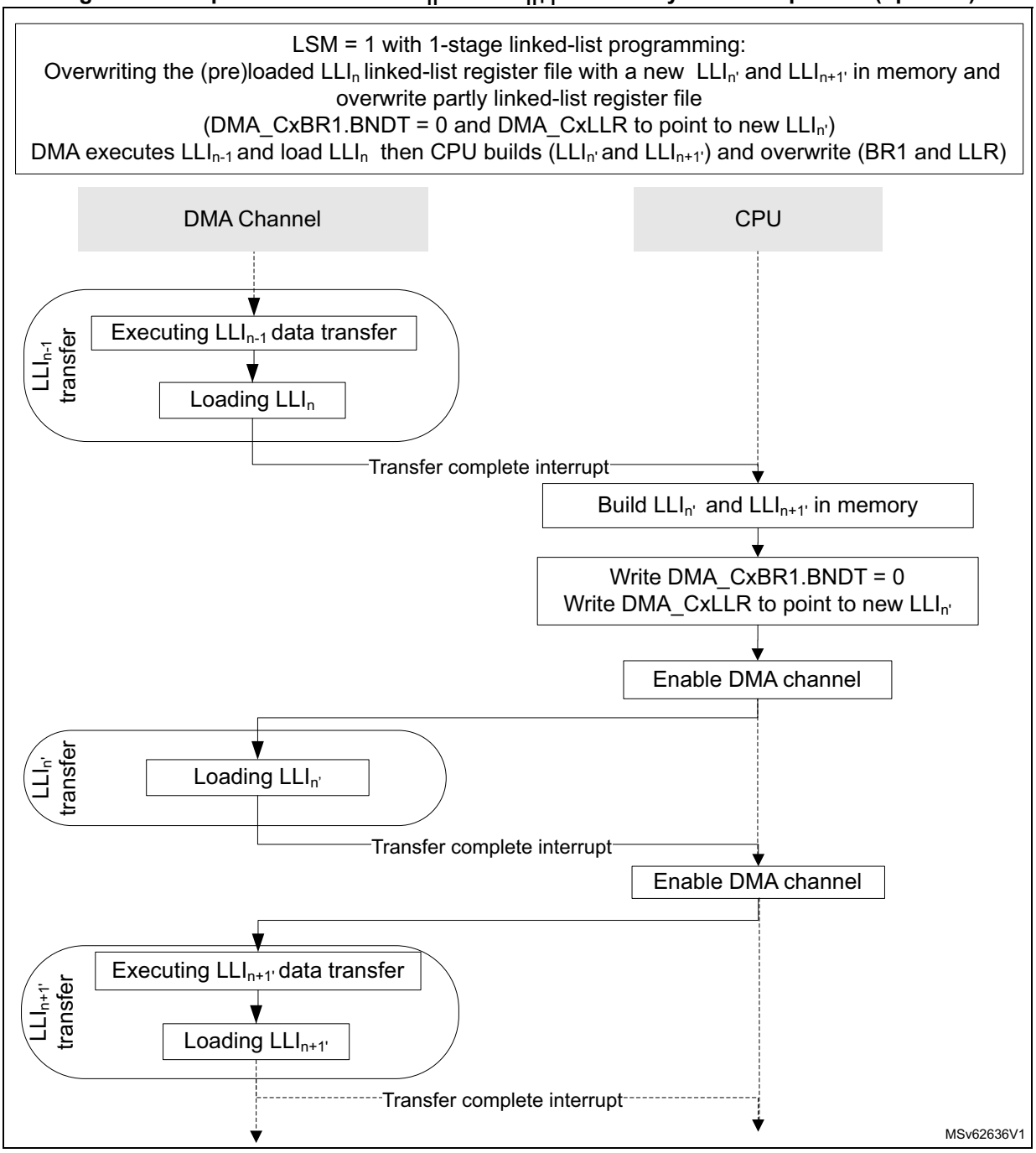
Figure 114. Replace with a new $LLI_{n'}$ in register file in link step mode



Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode (in the memory)

The software can build and insert a new $LLI_{n'}$ and $LLI_{n+1'}$ in the memory, after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory, by overwriting partly the linked-list register file (GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new $LLI_{n')}$ as shown in *Figure 115*.

Figure 115. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 1)



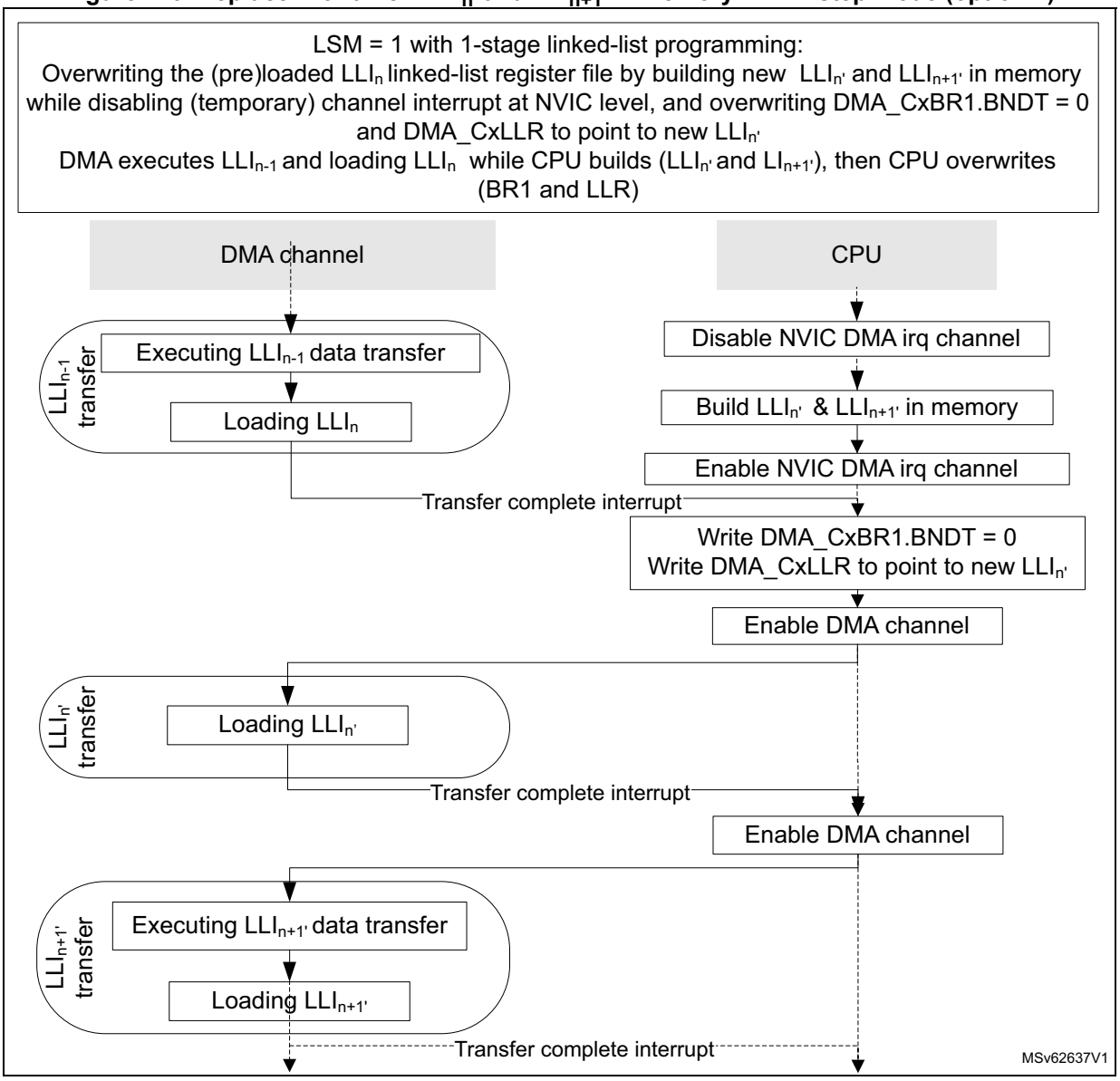
Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode

Other software implementations exist. Meanwhile GPDMA executes the transfer from the LLI_{n-1} and loads a formerly elaborated LLI_n from the memory (or even earlier), the software can do the following:

1. Disable the NVIC for not being interrupted by the interrupt handling.
2. Build a new $LLI_{n'}$ and a new $LLI_{n+1'}$.
3. Enable again the NVIC for the channel interrupt (transfer complete) notification.

The software in the interrupt handler for LLI_{n-1} is then restricted to overwrite GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new $LLI_{n'}$, as shown in *Figure 116*.

Figure 116. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 2)



13.4.10 GPDMA channel state and linked-list programming

The software can reconfigure a channel when the channel is disabled (GPDMA_CxCR.EN = 0) and update the execution mode (GPDMA_CxCR.LSM) to change from/to run-to-completion mode to/from link step mode.

In any execution mode, the software can:

- reprogram LLI_{n+1} in the memory to finally complete the channel by this LLI_{n+1} (clear the GPDMA_CxLLR of this LLI_{n+1}), before this LLI_{n+1} is loaded/used by the GPDMA channel
- abort and reconfigure the channel with a LSM update (see *Section 13.4.4*.)

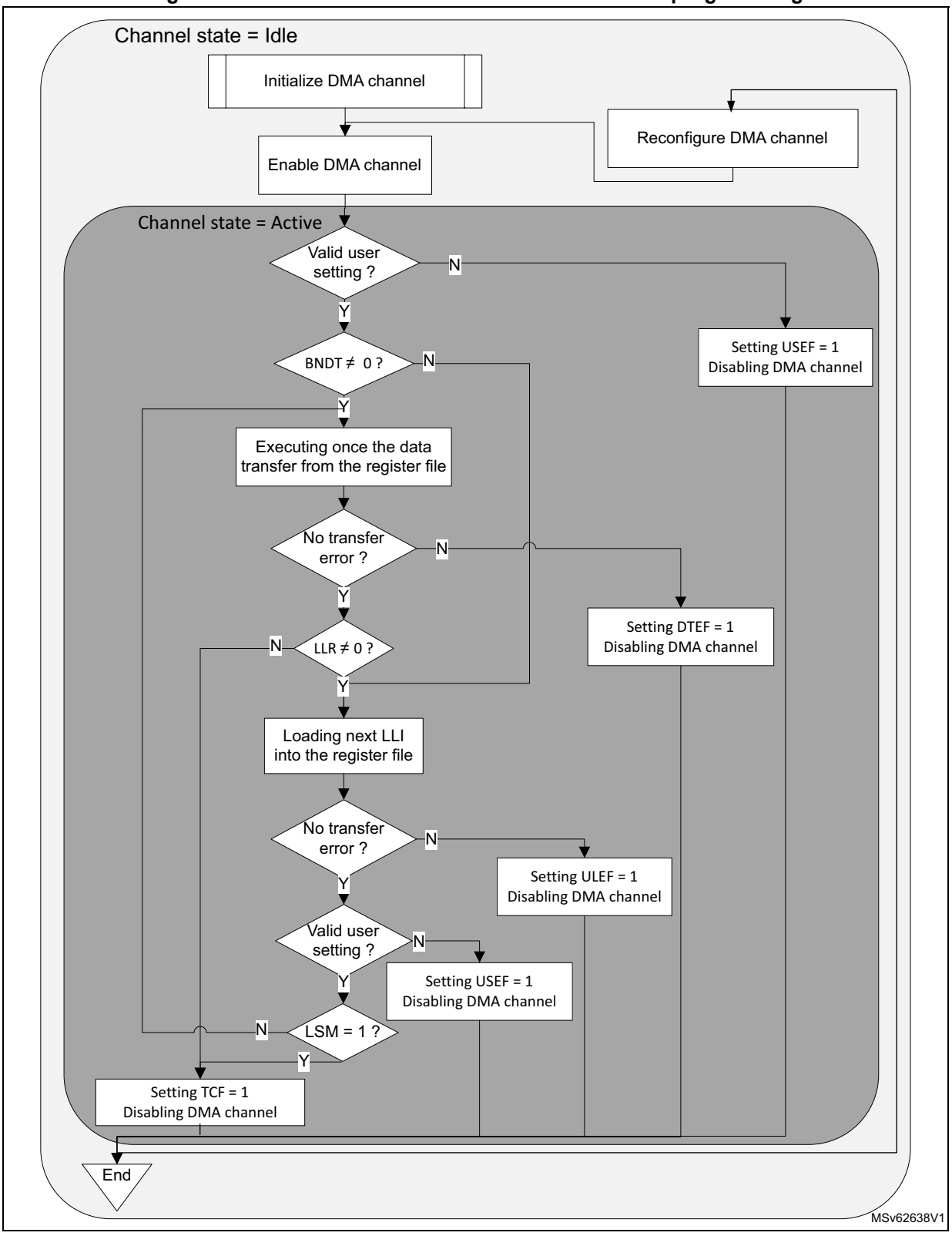
In link step mode, the software can clear LSM after each a single execution of any LLI, during LLI_{n-1}.

Figure 117 shows the overall and unified GPDMA linked-list programming, whatever is the execution mode.

Note: Figure 117 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion the hardware always set TCEF = 1 and disables the channel. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 117, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 13.3.5).

Figure 117. GPDMA channel execution and linked-list programming



13.4.11 GPDMA FIFO-based transfers

There is a single transfer operation mode: the FIFO mode. There are FIFO-based transfers. Any channel x is implemented with a dedicated FIFO whose size is defined by dma_fifo_size[x] (see *Section 13.3.1* for more details).

GPDMA burst

A programmed transfer at the lowest level is a GPDMA burst.

A GPDMA burst is a burst of data received from the source, or a burst of data sent to the destination. A source (and destination) burst is programmed with a burst length by the field SBL_1[5:0] (respectively DBL_1[5:0]), and with a data width defined by the field SDW_LOG2[1:0] (respectively DDW_LOG2[1:0]) in the GPDMA_CxTR1 register.

The addressing mode after each data (named beat) of a GPDMA burst is defined by SINC and DINC in GPDMA_CxTR1, for source and destination respectively: either a fixed addressing or an incremented addressing with contiguous data.

The start and next addresses of a GPDMA source/destination burst (defined by GPDMA_CxSAR and GPDMA_CxDAR) must be aligned with the respective data width.

The table below lists the main characteristics of a GPDMA burst.

Table 100. Programmed GPDMA source/destination burst

SDW_LOG2[1:0] DDW_LOG2[1:0]	Data width (bytes)	SINC/DINC	SBL_1[5:0] DBL_1[5:0]	Burst length (data/beats)	Next data/ beat address	Next burst address	Burst address alignment
00	1	0 (fixed)	n = 0 to 63(1)	n+1	+ 0	+ 0	1
01	2						2
10	4						4
00	1	1 (contiguously incremented)			+ 1	+ (n + 1)	1
01	2				+ 2	+ 2 * (n + 1)	2
10	4				+ 4	+ 4 * (n + 1)	4
11	forbidden user setting, causing USEF generation and none burst to be issued.						

1. When S/DBL_1[5:0] = 0, burst is of length 1. Then burst can be also named as single.

The next burst address in the above table is the next source/destination default address pointed by GPDMA_CxSAR or GPDMA_CxDAR, once the programmed source/destination burst is completed. This default value refers to the fixed/contiguously incremented address.

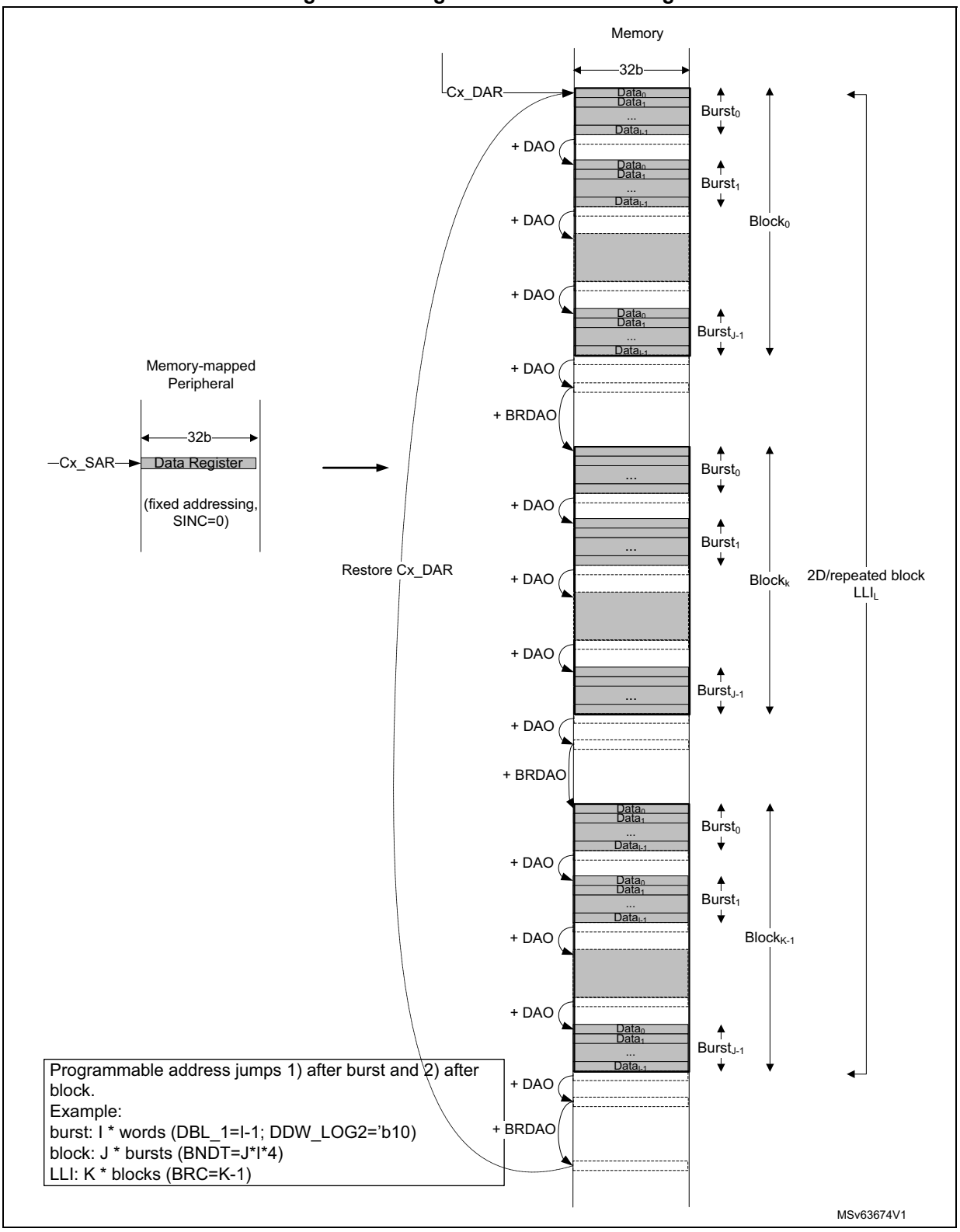
GPDMA burst with 2D addressing (channel x = 12 to 15)

When the channel has additional 2D addressing feature, this default value refers to the value without taking into account the two programmed incremented or decremented offsets. These two additional offsets (with a null default value) are applied:

- after each completed source/destination burst, as defined respectively by GPDMA_CxTR2.SAO[12:0]/DAO[12:0] and GPDMA_CxBR1.SDEC/DDEC
- after each completed block, as defined respectively by GPDMA_CxBR2.BRSAO[15:0]/BRDAO[15:0] and GPDMA_CxBR1.BRSDEC/BRDDEC)

Then, a 2D/repeated block can be addressed with a first programmed address jump after each completed burst, and with a second programmed address jump after each block, as depicted by *Figure 118* with a 2D destination buffer.

Figure 118. Programmed 2D addressing



GPDMA FIFO-based burst

In FIFO-mode, a transfer generally consists of two pipelined and separated burst transfers:

- one burst from the source to the FIFO over the allocated source master port, as defined by GPDMA_CxTR1.SAP
- one burst from the FIFO to the destination over the allocated destination master port, as defined by GPDMA_CxTR1.DAP

GPDMA source burst

The requested source burst transfer to the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to get one new burst from the source):

when FIFO level ≤ $2^{dma_fifo_size[x]}$ - (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- $2^{dma_fifo_size[x]}$ is the half the FIFO size of channel x, in bytes (see *Section 13.3.1* for the implementation details and dma_fifo_size[x] value).
- (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$ is the size of the programmed source burst transfer, in bytes.

Based on the channel priority (GPDMA_CxCR.PRIO[1:0]), this ready FIFO-based source transfer is internally arbitrated versus the other requested and active channels.

GPDMA destination burst

The requested destination burst transfer from the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to push one new burst to the destination):

when FIFO level ≥ (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$ is the size of the programmed destination burst transfer, in bytes.

Based on the channel priority, this ready FIFO-based destination transfer is internally arbitrated versus the other requested and active channels.

GPDMA burst vs source block size, 1-Kbyte address boundary and FIFO size

The programmed source/destination GPDMA burst is implemented with an AHB burst as is, unless one of the following conditions is met:

- When half of the FIFO size of the channel x is lower than the programmed source/destination burst size, the programmed source/destination GPDMA burst is implemented with a series of singles or bursts of a lower size. Each transfer is of a size that is lower than or equal to half of the FIFO size, without any user constraint.
- If the source block size (GPDMA_CxBR1.BNDT[15:0]) is not a multiple of the source burst size but is a multiple of the data width of the source burst (GPDMA_CxTR1.SDW_LOG2[1:0]), the GPDMA modifies and shortens bursts into singles or bursts of lower length, in order to transfer exactly the source block size, without any user constraint.

- if the source/destination burst transfer have crossed the 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.
- If the source/destination burst length exceeds 16 on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.

In any case, the GPDMA keeps ensuring source/destination data (and address) integrity without any user constraint. The current FIFO level (software readable in GPDMA_CxSR) is compared to and updated with the effective transfer size, and the GPDMA re-arbitrates between each AHB single or burst transfer, possibly modified.

Based on the channel priority, each single or burst of a lower burst size versus the programmed burst, is internally arbitrated versus the other requested and active channels.

Note: In linked-list mode, the GPDMA read transfers related to the update of the linked-list parameters from the memory to the internal GPDMA registers, are scheduled over the link allocated port, as programmed by GPDMA_CxCR.LAP.

GPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension and left/right alignment

The data handling is controlled by GPDMA_CxTR1. The source/destination data width of the programmed burst is byte, half-word or word, as per the SDW_LOG2[1:0] and DDW_LOG2[1:0] fields (see *Table 101*).

The user can configure the data handling between transferred data from the source and transfer to the destination. More specifically, programmed data handling is orderly performed with:

1. Byte-based source reordering
   - If SBX = 1 and if source data width is a word, the two bytes of the unaligned half-word at the middle of each source data word are exchanged.
2. Data width conversion by packing, unpacking, padding or truncation, if destination data width is different than the source data width, depending on PAM[1:0]:
   - If destination data width > source data width, the post SBX source data is either right-aligned and padded with 0 s, or sign extended up to the destination data width, or is FIFO queued and packed up to the destination data width.
   - If destination data width < source data width, the post SBX data is either right-aligned and left-truncated down to the destination data width, or is FIFO queued and unpacked and streamed down to the destination data width.
3. Byte-based destination re-ordering:
   - If DBX = 1 and if the destination data width is not a byte, the two bytes are exchanged within the aligned post PAM[1:0] half-words.
   - If DHX = 1 and if the destination data width is neither a byte nor a half-word, the two aligned half-words are exchanged within the aligned post PAM[1:0] words.

Note: Left-alignment with 0s-padding can be achieved by programming both a right-alignment with a 0s-padding and a destination byte-based re-ordering.

The table below lists the possible data handling from the source to the destination.

Table 101. Programmed data handling

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
00	Byte	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$	x	00	Byte	xx	x	x	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, 0P)	0		$0B_3,0B_2,0B_1,0B_0$
							1		B_30,B_20,B_10,B_00
						01 (RA, SE)	0		SB_3,SB_2,SB_1,SB_0
							1		B_3S,B_2S,B_1S,B_0S
						1x (PACK)	0		$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)	0	0	$000B_1,000B_0$
							1		$00B_10,00B_00$
							0	1	$0B_100,0B_000$
							1		B_1000,B_0000
						01 (RA, SE)	0	0	$SSSB_1,SSSB_0$
							1		SSB_1S,SSB_0S
							0	1	SB_1SS,SB_0SS
							1		B_1SSS,B_0SSS
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$		00	Byte	00 (RA, LT)	x	x	B_6,B_4,B_2,B_0
						01 (LA, RT)			B_7,B_5,B_3,B_1
						1x (UNPACK)			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$

Table 101. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$	x	01	Half-word	xx	0	x	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)	0	0	$00B_3B_2,00B_1B_0$
							1		$00B_2B_3,00B_0B_1$
							0	1	B_3B_200,B_1B_000
							1		B_2B_300,B_0B_100
						01 (RA, SE)	0	0	SSB_3B_2,SSB_1B_0
							1		SSB_2B_3,SSB_0B_1
							0	1	B_3B_2SS,B_1B_0SS
							1		B_2B_3SS,B_0B_1SS
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
10	Word	$B_7B_6B_5B_4,B_3B_2B_1B_0$	0	00	Byte	00 (RA, LT)	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)			B_{15},B_{11},B_7,B_3
						10 (UNPACK)			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, LT)	0		B_5B_4,B_1B_0
							1		B_4B_5,B_0B_1
						01 (LA, RT)	0		B_7B_6,B_3B_2
							1		B_6B_7,B_2B_3
						1x (UNPACK)	0		$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$

Table 101. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	0	10	Word	xx	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
			1	00	Byte	00 (RA, LT)	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)			B_{15},B_{11},B_7,B_3
						1x (UNPACK)			$B_7,B_5,B_6,B_4,B_3,B_1,B_2,B_0$
				01	Half-word	00 (RA, LT)	0		B_6B_4,B_2B_0
							1		B_4B_6,B_0B_2
						01 (LA, RT)	0		B_7B_5,B_3B_1
							1		B_5B_7,B_1B_3
						1x (UNPACK)	0		$B_7B_5,B_6B_4,B_3B_1,B_2B_0$
							1		$B_5B_7,B_4B_6,B_1B_3,B_0B_2$
				10	Word	xx	0	0	$B_7B_5B_6B_4,B_3B_1B_2B_0$
							1		$B_5B_7B_4B_6,B_1B_3B_0B_2$
							0	1	$B_6B_4B_7B_5,B_2B_0B_3B_1$
							1		$B_4B_6B_5B_7,B_0B_2B_1B_3$

1. Data stream is timely ordered starting from the byte with the lowest index (B_0).
2. RA= right aligned, LA = left aligned, RT = right truncated, LT = left truncated, 0P = zero bit padding up to the destination data width, SE = sign bit extended up to the destination data width.

13.4.12 GPDMA transfer request and arbitration

GPDMA transfer request

As defined by GPDMA_CxTR2, a programmed GPDMA data transfer is requested with one of the following:

- a software request if the control bit SWREQ = 1: This is used typically by the CPU for a data transfer from a memory-mapped address to another memory mapped address (memory-to-memory, GPIO to/from memory)
- an input hardware request coming from a peripheral if SWREQ = 0: The selection of the GPDMA hardware peripheral request is driven by the REQSEL[7:0] field (see *Section 13.3.3*). The selected hardware request can be one of the following:
  - an hardware request from a peripheral configured in GPDMA mode (for a transfer from/to the peripheral data register respectively to/from the memory)
  - an hardware request from a peripheral for its control registers update from the memory
  - an hardware request from a peripheral for a read of its status registers transferred to the memory

Caution: The user must not assign a same input hardware peripheral GPDMA request via GPDMA_CxTR.REQSEL[7:0] to two different channels, if at a given time this request is asserted by the peripheral and each channel is ready to execute this requested data transfer. There is no user setting error reporting.

GPDMA transfer request for arbitration

A ready FIFO-based GPDMA source single/burst transfer (from the source address to the FIFO) to be scheduled over the allocated master port (GPDMA_CxTR1.SAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

A ready FIFO-based GPDMA destination single/burst transfer (from the FIFO to the destination address) to be scheduled over the allocated master port (GPDMA_CxTR1.DAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

An arbitrated GPDMA requested link transfer consists of one 32-bit read from the linked-list data structure in memory to one of the linked-list registers (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR or GPDMA_CxLLR, plus GPDMA_CxTR3, GPDMA_CxBR2). Each 32-bit read from memory is arbitrated with the same channel priority as for data transfers, in order to be scheduled over the allocated master port (GPDMA_CxCR.LAP).

Whatever the requested data transfer is programmed with a software request for a memory-to- memory transfer (GPDMA_CxTR2.SWREQ = 1), or with a hardware request (GPDMA_CxTR2.SWREQ = 0) for a memory-to-peripheral transfer or a peripheral-to-memory transfer and whatever is the hardware request type, re-arbitration occurs after each granted single/burst transfer.

When an hardware request is programmed from a destination peripheral (GPDMA_CxTR2.SWREQ = 0 and GPDMA_CxTR2.DREQ = 1), the first memory read of a (possibly 2D/repeated) block (the first ready FIFO-based source burst request), is gated by the occurrence of the corresponding and selected hardware request. This first read request to memory is not taken into account earlier by the arbiter (not as soon as the block transfer is enabled and executable).

GPDMA arbitration

The GPDMA arbitration is directed from the 4-grade assigned channel priority (GPDMA_CxCR.PRIO[1:0]). The arbitration policy, as illustrated in *Figure 119*, is defined by:

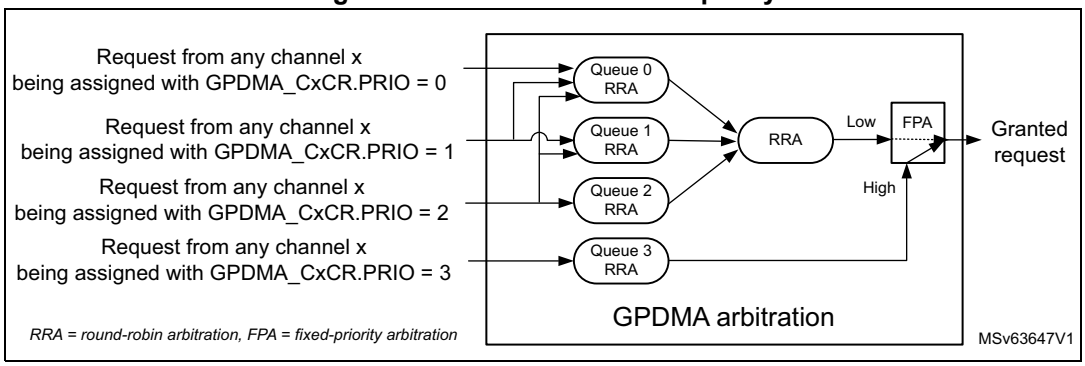
- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels

  This traffic class is granted via a fixed-priority arbitration against any other low-priority traffic class. Within this class, requested single/burst transfers are round-robin arbitrated.
- three low-priority traffic classes (queues 0, 1, or 2) for non time-sensitive channels with priority 0, 1, or 2

  Each requested single/burst transfer within this class is round-robin arbitrated, with a weight that is monotonically driven from the programmed priority:
  - Requests with priority 0 are allocated to the queue 0.
  - Requests with priority 1 are allocated and replicated to the queue 0 and queue 1.

– Requests with priority 2 are allocated and replicated to the queue 0, queue 1, and queue 2.
– Any queue 0, 1, or 2 equally grants any of its active input requests in a round-robin manner, provided there are simultaneous requests.
– Additionally, there is a second stage for the low-traffic with a round-robin arbiter that fairly alternates between simultaneous selected requests from queue 0, queue 1, and queue 2.

Figure 119. GPDMA arbitration policy



GPDMA arbitration and bandwidth

With this arbitration policy, the following is guaranteed:

- Equal maximum bandwidth between requests with same priority
- Reserved bandwidth (noted as B_{Q3}) to the time-sensitive requests (with priority 3)
- Residual weighted bandwidth between different low-priority requests (priority 0 versus priority 1 versus priority 2).

The two following examples highlight that the weighted round-robin arbitration is driven by the programmed priorities:

- **Example 1**: basic application with two non time-sensitive GPDMA requests: req0 and req1. There are the following programming possibilities:
  – If they are assigned with same priority, the allocated bandwidth by the arbiter to req0 (B_{req0}) is **equal** to the allocated bandwidth to req1(B_{req1}).
  $B_{req0} = B_{req1} = 1/2 * (1 - B_{Q3})$
  – If req0 is assigned to priority 0 and req1 to priority 1, the allocated bandwidth to req0 (B_{P0}) is **3 times less** than the allocated bandwidth to req1 (B_{P1}).
  $B_{req0} = B_{P0} = 1/2 * 1/2 * (1 - B_{Q3}) = 1/4 * (1 - B_{Q3})$
  $B_{req1} = B_{P1} = (1/2 + 1) * 1/2 * (1 - B_{Q3}) = 3/4 * (1 - B_{Q3})$
  – If req0 is assigned to priority 0 and req1 to priority 2, the allocated bandwidth to req0 (B_{P0}) is **5 times less** than the allocated bandwidth to req1 (B_{P2}).
  $B_{req0} = B_{P0} = 1/2 * 1/3 * (1 - B_{Q3}) = 1/6 * (1 - B_{Q3})$
  $B_{req1} = B_{P2} = (1/2 + 1 + 1) * 1/3 * (1 - B_{Q3}) = 5/6 * (1 - B_{Q3})$
  The above computed bandwidth calculation is based on a theoretical input request, always active for any GPDMA clock cycle. This computed bandwidth from the arbiter must be weighted by the frequency of the request given by the application, that cannot be always active and may be quite much variable from one GPDMA client (example I2C at 400 kHz) to another one (PWM at 1 kHz) than the above x3 and x5 ratios.

- **Example 2**: application where the user distributes a same non-null N number of GPDMA requests to every non time-sensitive priority 0, 1 and 2. The bandwidth calculation is then the following:
  - The allocated bandwidth to the set of requests of priority 0 (B_{P0}) is
    $B_{P0} = 1/3 * 1/3 * (1 - B_{Q3}) = 1/9 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 1(B_{P1}) is
    $B_{P1} = (1/3 + 1/2) * 1/3 * (1 - B_{Q3}) = 5/18 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 2(B_{P2}) is
    $B_{P2} = (1/3 + 1/2 + 1) * 1/3 * (1 - B_{Q3}) = 11/18 * (1 - B_{Q3})$
  - The allocated bandwidth to any request n ($B_{n)}$ among the N requests of that priority Pi (i = 0 to 2) is $B_n = 1/N * B_{Pi}$
  - The allocated bandwidth to any request n of priority 0i ($B_{n, Pi}$) is
    $B_{n, P0} = 1/N *1/9 * (1 - B_{Q3})$
    $B_{n, P1} = 1/N *5/18 * (1 - B_{Q3})$
    $B_{n, P2} = 1/N *11/18 * (1 - B_{Q3})$

  In this example, when the master port bus bandwidth is not totally consumed by the time-sensitive queue 3, the residual bandwidth is such that 2.5 times less bandwidth is allocated to any request of priority 0 versus priority 1, and 5.5 times less bandwidth is allocated to any request of priority 0 versus priority 2.

More generally, assume that the following requests are present:

- I requests (I ≥ 0) assigned to priority 0
  If I > 0, these requests are noted from i = 0 to I-1.
- J requests (J ≥ 0) assigned to priority 1
  If J > 0, these requests are noted from j = 0 to J-1.
- K requests (K > 0) assigned to priority 2
  These requests are noted from k = 0 to K-1
- L requests (L ≥ 0) assigned to priority 3
  If L > 0, these requests are noted from l = 0 to L-1.

As B_{Q3} is the reserved bandwidth to time-sensitive requests, the bandwidth for each request L with priority 3 is:

- $B_l = B_{Q3} / L$ for L > 0 (else: $B_l = 0$)

The bandwidth for each non-time sensitive queue is:

- $B_{Q0} = 1/3 * (1 - B_{Q3})$
- $B_{Q1} = 1/3 * (1 - B_{Q3})$
- $B_{Q2} = 1/3 * (1 - B_{Q3})$

The bandwidth for the set of requests with priority 0 is:

- $B_{P0} = I / (I + J + K) * B_{Q0}$

The bandwidth for each request i with priority 0 is:

- $B_i = B_{P0} / I$ for L > 0 (else $B_{P0} = 0$)

The bandwidth for the set of requests with priority 1 and routed to queue 0 is:

- $B_{P1,Q0} = J / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 1 and routed to queue 1 is:

- $B_{P1,Q1} = J / (J + K) * B_{Q1}$

The total bandwidth for the set of requests with priority 1 is:

- $B_{P1} = B_{P1,Q0} + B_{P1,Q1}$

The bandwidth for each request j with priority 1 is:

- $B_j = B_{P1} / J$ for $J > 0$ (else $B_j = 0$)

The bandwidth for the set of requests with priority 2 and routed to queue 0 is:

- $B_{P2,Q0} = K / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 2 and routed to queue 1 is:

- $B_{P2,Q1} = K / (J + K) * B_{Q1}$

The bandwidth for the set of requests with priority 2 and routed to queue 2 is:

- $B_{P2,Q2} = B_{Q2}$

The total bandwidth for the set of requests with priority 2 is:

- $B_{P2} = B_{P2,Q0} + B_{P2,Q1} + B_{P2,Q2}$

The bandwidth for each request k with priority 2 is:

- $B_k = B_{P2} / K$ ($K>0$ in the general case)

Thus finally the maximum allocated residual bandwidths for any i, j, k non-time sensitive request are:

- in the general case (when there is at least one request k with a priority 2 ($K > 0$)):
  - $B_i = 1/I * 1/3 * I/(I + J + K) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/3 *[J/(I + J + K) + J/(J + K)] * (1 - B_{Q3})$
  - $B_k = 1/K * 1/3 *[K/(I + J + K) + K/(J + K) + 1] * (1 - B_{Q3})$
- in the specific case (when there is no request k with a priority 2 ($K = 0$)):
  - $B_i = 1/I * 1/2 * I/(I + J) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/2 *[J/(I + J) + 1] * (1 - B_{Q3})$

Consequently, the GPDMA arbiter can be used as a programmable weighted bandwidth limiter, for each queue and more generally for each request/channel. The different weights are monotonically resulting from the programmed channel priorities.

13.4.13 GPDMA triggered transfer

A programmed GPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event, as defined by GPDMA_CxTR2.TRIGPOL[1:0] and GPDMA_CxTR2.TRIGSEL[5:0] (see *Section 13.3.6* for the trigger selection).

The triggered transfer, as defined by the trigger mode in GPDMA_CxTR2.TRIGM[1:0], can be at LLI data transfer level, to condition the first burst read of a block, the first burst read of a 2D/repeated block for channel x (x = 12 to 15), or each programmed single read. The trigger mode can also be programmed to condition the LLI link transfer (see TRIGM[1:0] in GPDMA_CxTR2 for more details).

Trigger hit memorization and trigger overrun flag generation

The GPDMA monitoring of a trigger for a channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (respectively TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

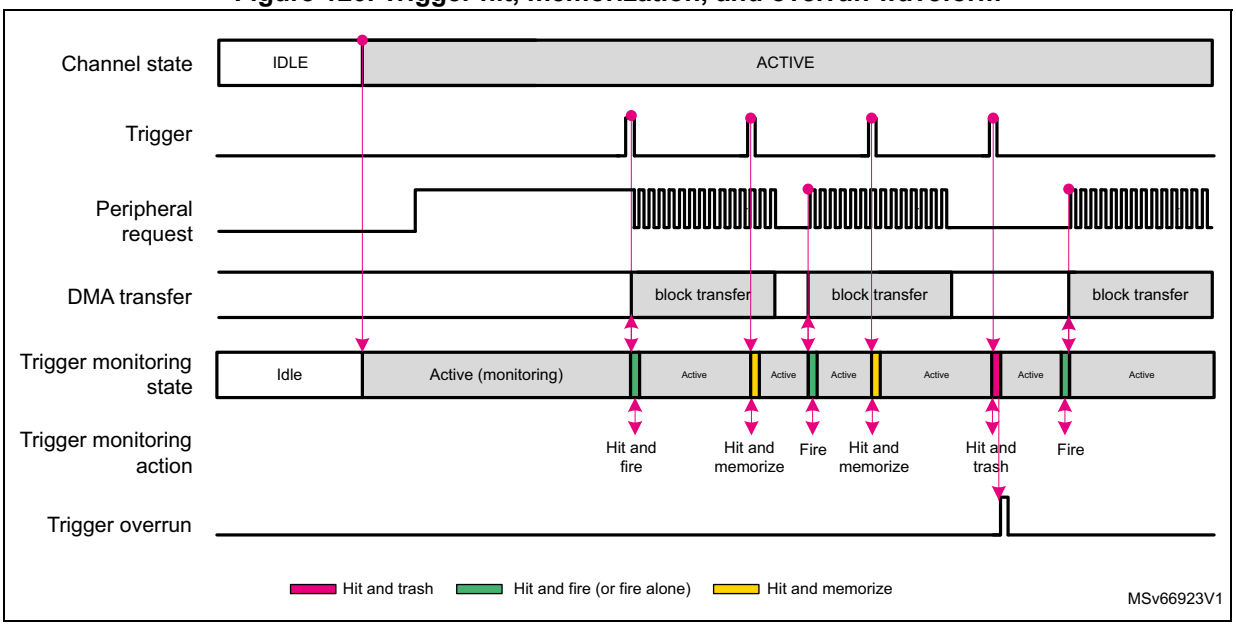
The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[5:0] are not modified, and the channel is enabled.

Transferring a next $LLI_{n+1,}$ that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the possible memorized hit of the formerly defined LLI_n trigger.

Caution: After a first new trigger hit_{n+1} is memorized, if another trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (GPDMA_CxSR.TOF = 1) and an interrupt is generated if enabled (if GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Figure 120 illustrates the trigger hit, memorization and overrun in the configuration example with a block-level trigger mode and a rising edge trigger polarity.

Figure 120. Trigger hit, memorization, and overrun waveform



Note: The user can assign the same input trigger event to different channels. This can be used to trigger different channels on a broadcast trigger event.

13.4.14 GPDMA circular buffering with linked-list programming

GPDMA circular buffering for memory-to-peripheral and peripheral-to-memory transfers, with a linear addressing channel

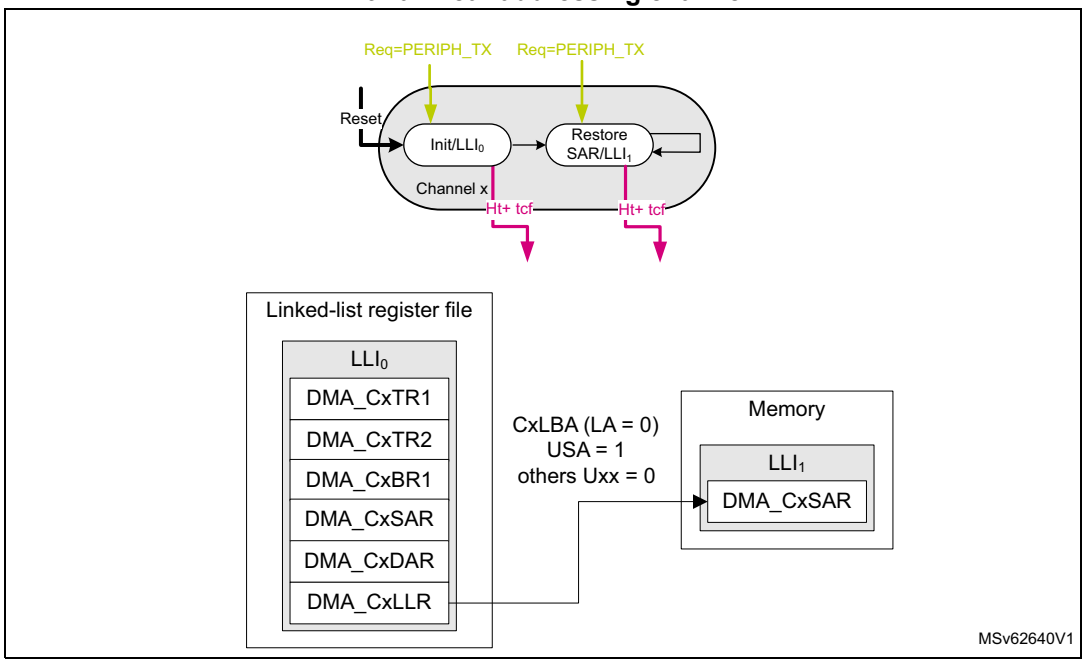
For a circular buffering, with a continuous memory-to-peripheral (or peripheral-to-memory) transfer, the software must set up a channel with half-transfer and complete transfer

events/interrupts generation (GPDMA_CxCR.HTIE = 1 and GPDMA_CxCR.TCIE = 1), in order to enable a concurrent buffer software processing.

LLI_0 is configured for the first block transfer with the linear addressing channel. A continuously-executed LLI_1 is needed to restore the memory source (or destination) start address, for the memory-to-peripheral transfer (respectively the peripheral-to-memory transfer). GPDMA automatically reloads the initially programmed GPDMA_CxBR1.BNDT[15:0] when a block transfer is completed, and there is no need to restore GPDMA_CxBR1.

Figure 121 illustrates this programming with a linear addressing GPDMA channel and a source circular buffer.

Figure 121. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel

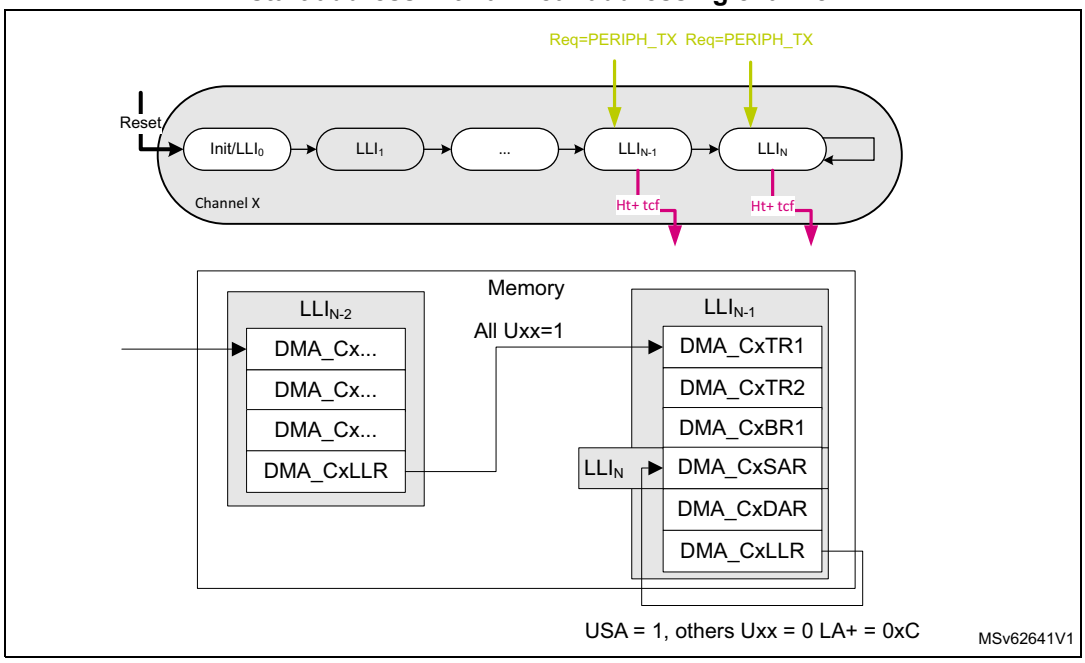


Note: With a 2D addressing channel, the user may use a single LLI with GPDMA_CxBR1.BRC[10:0] = 1, and program a negative memory block address offset with GDMA_CxBR2 and GDMA_CxBR1, in order to jump back to the memory source or the destination start address.

If circular buffering must be executed after some other transfers over the shared GPDMA channel x, the before-last LLI_{N-1} in memory is needed to configure the first block transfer. And the last LLI_N restores the memory source (or destination) start address in memory-to-peripheral transfer (respectively in peripheral-to-memory transfer).

Figure 122 illustrates this programming with a linear addressing shared GPDMA channel, and a source circular buffer.

Figure 122. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel



13.4.15 GPDMA transfer in peripheral flow-control mode

A peripheral with the peripheral flow-control mode feature can decide to early terminate a GPDMA block transfer, provided that the allocated channel is implemented with this feature (see *Section 13.3.5*).

If the related GPDMA channel x is also programmed in peripheral flow-control mode (GPDMA_CxTR2.PFREQ = 1):

- The GPDMA block transfer starts as follows:
  - If GPDMA_CxBR1.BNDT[15:0] ≠ 0, the programmed value is internally taken into account by the GPDMA hardware.
  - If GPDMA_CxBR1.BNDT[15:0] = 0, the GPDMA hardware internally considers a 64-Kbyte value for the maximum source block size to be transferred.
- The GPDMA block transfer is completed as soon as the first occurrence of one of the following condition occurs:
  - when GPDMA_CxBR1.BNDT[15:0] = 0
  - when the peripheral early terminates the block. The complete transfer event is generated if programmed, depending on GPDMA_CxTR2 (see *GPDMA channel x transfer register 2 (GPDMA_CxTR2)*). Then the software can read the current number of transferred bytes from the source (GPDMA_CxBR1.BNDT[15:0]), and/or read the current source or destination address of the buffer in memory (GPDMA_CxSAR[31:0] or GPDMA_CxDAR[31:0]).

In peripheral flow-control mode:

- a destination peripheral with a hardware requested transfer is not supported: memory-to-peripheral transfer is not supported.
- Data packing from a source peripheral is not supported.
- 2D/repeated block is not supported.
- GPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

13.4.16 GPDMA byte-based padding at the block level

For a given channel x, a LLI_n data transfer consists of a block transfer and is defined by the source block size GPDMA_CxBR1.BNDT[15:0], the size of the block of data to be transfered from the source, in bytes.

During a peripheral-to-memory transfer, a peripheral may be streaming and receiving a given number of bytes (for example, via an I2C, UART/USART/LPUART, ADC, DAC) or half-words, and this block size may not be a multiple of a word or a double-word. It is important to allow the user to program any number of bytes for the GPDMA source block size independently from the source data width and the destination data width, in order to use efficiently the bandwidth of the 32-bit AHB system bus and/or the 64-bit AXI system bus when accessing and writing into memory (embedded SRAM, NVM or external memory). This enables to decrease the bus bandwidth for the memory transfers by a ratio of up to 4 in AHB and up to 8 in AXI. More generally, having the source block size independent from the source data width and the destination data width can be useful also with memory-to-peripheral transfers, and with memory-to-memory transfers (AHB from/to AXI).

A FIFO-based channel is able to perform byte-based padding on the last write(s) up to the fully transferred source block size. Then channel FIFO is empty. Data handling is performed on zero bytes in the same way as on regular data bytes from the source, as explained in *GPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension and left/right alignment*.

There is byte-based padding at the block level:

1. If source data width is less than the destination data width, and if FIFO queuing and data packing is configured (PAM[1:0] = 1x):

- Padding with zero bytes. For example, if BNDT= 5, 8-bit reads, 32-bit writes:
  - if DBX = 0, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> 000B_4, $B_3B_2B_1B_0$
  - if DBX = 0, DHX = 1: B_4,B_3,B_2,B_1,B_0 -> 0$B_4$00, $B_1B_0B_3B_2$
  - if DBX = 1, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> 00$B_4$0, $B_2B_3B_0B_1$
  - if DBX = 1, DHX = 1: B_4,B_3,B_2,B_1,B_0 -> $B_4$000, $B_0B_1B_2B_3$

2. If source data width > destination data width, and if FIFO queuing and data unpacking is configured (PAM[1:0] = 1x):

- Padding with zero bytes.
- On the last read, only the number of LSB(s) up to the block size are written to the FIFO. The last written data to the destination is padded with zero bytes up to the destination data width. For example, if BNDT= 5, 32-bit reads, 16-bit writes:
  - if DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 0B_4, B_3B_2, B_1B_0
  - if DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$, -> $B_4$0, B_2B_3, B_0B_1

3. If source data width = destination data width (and BNDT[15:0] is not a multiple of that value):

- Padding with zero bytes. For example, if BNDT= 6, 32-bit reads, 32-bit writes are configured
  - if SBX = 0, DBX = 0, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $00B_5B_4$, $B_3B_2B_1B_0$
  - if SBX = 0, DBX = 0, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_5B_400, $B_1B_0B_3B_2$
  - if SBX = 0, DBX = 1, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $00B_4B_5$, $B_2B_3B_0B_1$
  - if SBX = 0, DBX = 1, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_4B_500, $B_0B_1B_2B_3$
  - if SBX = 1, DBX = 0, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $000B_4$, $B_3B_1B_2B_0$
  - if SBX = 1, DBX = 0, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $0B_400$, $B_2B_0B_3B_1$
  - if SBX = 1, DBX = 1, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $00B_40$, $B_1B_3B_0B_2$
  - if SBX = 1, DBX = 1, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_4000, $B_0B_2B_1B_3$

4. If source data width < destination data width, and if destination data is configured as right-aligned and, either configured with zero padding bytes (PAM[1:0] = 00), or configured with sign extension (PAM[1:0] = 01):

- Padding with zero bytes if PAM[1:0] = 00. For example, BNDT = 5, 8-bit reads, 32-bit writes, PAM[1:0] = 00:
  - if DBX =0, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> $000B_4$, $000B_3$, $000B_2$, $000B_1$, $000B_0$
- Padding with sign extension if PAM[1:0] = 01. For example, BNDT = 5, 8-bit reads, 32-bit writes, PAM[1:0] = 01:
  - if DBX = 0, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> $SSSB_4$, $SSSB_3$, $SSSB_2$, $SSSB_1$, $SSSB_0$

5. If source data width > destination data width, and if destination data is configured as right-aligned (PAM[1:0] = 00), or left-aligned (PAM[1:0] = 01):

- Padding with zero bytes. For example, BNDT = 5, 32-bit reads, 16-bit writes:
  - if SBX = 0, PAM[1:0] = 00, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $0B_4$, B_1B_0
  - if SBX = 0, PAM[1:0] = 00, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_40, B_0B_1
  - if SBX = 0, PAM[1:0] = 01, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_3B_2
  - if SBX = 0, PAM[1:0] = 01, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_2B_3
  - if SBX = 1, PAM[1:0] = 00, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $0B_4$, B_2B_0
  - If SBX = 1, PAM[1:0] = 00, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_40, B_0B_2
  - If SBX = 1, PAM[1:0] = 01, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_3B_1
  - If SBX = 0, PAM[1:0] = 01, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_1B_3

13.4.17 GPDMA privileged/unprivileged channel

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged agent via GPDMA_PRIVCFGR.PRIVx.

When a channel x is configured in a privileged state by a privileged agent, the following access control rules are applied:

- An unprivileged read access to a register field of this channel is forced to return 0, except for GPDMA_PRIVCFGR GPDMA_RCFGLOCKR that is readable by an unprivileged agent.
- An unprivileged write access to a register field of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) state, the source and destination data transfers are privileged (respectively unprivileged) transfers over the AHB master port.

When a channel is configured in a privileged (or unprivileged) state and in linked-list mode, the loading of the next linked-list data structure from the GPDMA memory into its register file, is automatically performed with privileged (respectively unprivileged) transfers, via the GPDMA_CxCR.LAP allocated master port.

The GPDMA generates a privileged bus that reflects GPDMA_PRIVCFGR, to keep the other peripherals informed of the privileged/unprivileged state of each GPDMA channel x.

When the privileged software must switch a channel from a privileged state to an unprivileged state, the privileged software must abort the channel or wait until that the privileged channel is completed before switching. This is needed to dynamically re-allocate a channel to a next unprivileged transfer as an unprivileged software is not allowed to do so, and must have GPDMA_CxCR.EN = 0 before the unprivileged software can reprogram the GPDMA_CxCR for a next transfer. The privileged software may reset not only the channel x (GPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

13.4.18 GPDMA error management

The GPDMA is able to manage and report to the user a transfer error, as follows, depending on the root cause.

Data transfer error

On a bus access (as a AHB single or a burst) to the source or the destination:

- The source or destination target reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware). The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the data error (GPDMA_CxSR.DTEF = 1).
- After a GPDMA data transfer error, the user must perform a debug session, taking care of the product-defined memory mapping of the source and destination, including the protection attributes.
- After a GPDMA data transfer error, the user must issue a channel reset (set GPDMA_CxCR.RESET) to reset the hardware GPDMA channel data path and the content of the FIFO, before the user enables again the same channel for a next transfer.

Link transfer error

On a tentative update of a GPDMA channel register from the programmed LLI in the memory:

- The linked-list memory reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware), the channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the link error (GPDMA_CxSR.ULEF = 1).
- After a GPDMA link error, the user must perform a debug session, taking care of the product-defined memory mapping of the linked-list data structure (GPDMA_CxLBAR and GPDMA_CxLLR), including the protection attributes.
- After a GPDMA link error, the user must explicitly write the linked-list register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR,

GPDMA_CxDAR and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2), before the user enables again the same channel for a next transfer.

User setting error

On a tentative execution of a GPDMA transfer with an unauthorized user setting:

- The programmed channel transfer is disabled (GPDMA_CxCR.EN forced and cleared by the GPDMA hardware) preventing the next unauthorized programmed data transfer from being executed. The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and a user setting error (GPDMA_CxSR.USEF = 1).
- After a GPDMA user setting error, the user must perform a debug session, taking care of the GPDMA channel programming. A user setting error can be caused by one of the following:
  - a programmed null source block size without a programmed update of this value from the next LLI_1 (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxLLR.UB1 = 0)
  - a programmed unaligned source start address, being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxSAR[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x (x = 12 to 15): a programmed unaligned source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxTR3.SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x (x = 12 to 15): a programmed unaligned block repeated source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxBR2.BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - a programmed unaligned destination start address, being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxDAR[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - for channel x (x = 12 to 15): a programmed unaligned destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxTR3.DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - for channel x (x = 12 to 15): a programmed unaligned block repeated destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxBR2.BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - a programmed linked-list item LLI_{n+1} with a null data transfer (GPDMA_CxLLR.UB1 = 1 and GPDMA_CxBR1. BNDT = 0)

13.5 GPDMA in debug mode

When the microcontroller enters debug mode (core halted), any channel x can be individually either continued (default) or suspended, depending on the programmable control bit in the DBGMCU module.

Note: In debug mode, GPDMA_CxSR.SUSPF is not altered by a suspension from the programmable control bit in the DBGMCU module. In this case, GPDMA_CxSR.IDLEF can be checked to know the completion status of the channel suspension.

13.6 GPDMA in low-power modes

Table 102. Effect of low-power modes on GPDMA

Mode	Description
Sleep	No effect. GPDMA interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the GPDMA registers is kept when entering Stop mode.
Standby	The GPDMA is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 13.3.2* to know if any Stop mode is supported.

13.7 GPDMA interrupts

There is one GPDMA interrupt line for each channel, and separately for each CPU (if several ones in the devices).

Table 103. GPDMA interrupt requests

Interrupt acronym	Interrupt event	Interrupt enable	Event flag	Event clear method
GPDMA_CHx	Transfer complete	GPDMA_CxCR.TCIE	GPDMA_CxSR.TCF	Write 1 to GPDMA_CxFCR.TCF
	Half transfer	GPDMA_CxCR.HTIE	GPDMA_CxSR.HTF	Write 1 to GPDMA_CxFCR.HTF
	Data transfer error	GPDMA_CxCR.DTEIE	GPDMA_CxSR.DTEF	Write 1 to GPDMA_CxFCR.DTEF
	Update link error	GPDMA_CxCR.ULEIE	GPDMA_CxSR.ULEF	Write 1 to GPDMA_CxFCR.ULEF
	User setting error	GPDMA_CxCR.USEIE	GPDMA_CxSR.USEF	Write 1 to GPDMA_CxFCR.USEF
	Suspended	GPDMA_CxCR.SUSPIE	GPDMA_CxSR.SUSPF	Write 1 to GPDMA_CxFCR.SUSPF
	Trigger overrun	GPDMA_CxCR.TOFIE	GPDMA_CxSR.TOF	Write 1 to GPDMA_CxFCR.TOF

A GPDMA channel x event may be:

- a transfer complete
- a half-transfer complete
- a transfer error, due to either:
  - a data transfer error
  - an update link error
  - a user setting error completed suspension
- a trigger overrun

Note: When a channel x transfer complete event occurs, the output signal gpdma_chx_tc is generated as a high pulse of one clock cycle.

An interrupt is generated following any xx event, provided that both:

- the corresponding interrupt event xx is enabled (GPDMA_CxCR.xxIE = 1)
- the corresponding event flag is cleared (GPDMA_CxSR.xxF = 0). This means that, after a previous same xx event occurrence, a software agent must have written 1 into the corresponding xx flag clear control bit (write 1 into GPDMA_CxFCR.xxF).

TCF (transfer complete) and HTF (half transfer) events generation is controlled by GPDMA_CxTR2.TCEM[1:0] as follows:

- A transfer complete event is a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].
- A half-transfer event is a half-block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].

  A half-block transfer occurs when half of the source block size bytes (rounded-up integer of GPDMA_CxBR1.BNDT[15:0] / 2) is transferred to the destination.

  A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded-up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 2) is transferred to the destination.

See *GPDMA channel x transfer register 2 (GPDMA_CxTR2)* for more details.

A transfer error rises in one of the following situations:

- during a single/burst data transfer from the source or to the destination (DTEF)
- during an update of a GPDMA channel register from the programmed LLI in memory (ULEF)
- during a tentative execution of a GPDMA channel with an unauthorized setting (USEF)

  The user must perform a debug session to correct the GPDMA channel programming versus the USEF root causes list (see *Section 13.4.18*).

A trigger overrun is described in *Trigger hit memorization and trigger overrun flag generation*.

13.8 GPDMA registers

The GPDMA registers must be accessed with an aligned 32-bit word data access.

13.8.1 GPDMA privileged configuration register (GPDMA_PRIVCFGR)

Address offset: 0x04

Reset value: 0x0000 0000

A write access to this register must be privileged. A read access can be privileged or unprivileged.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN =1.

This register must be programmed at a bit level, at the initialization/closure of a GPDMA channel (GPDMA_CxCR.EN = 0), to individually allocate any channel x to the privileged or unprivileged world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PRIVx**: privileged state of channel x (x = 15 to 0)
0: unprivileged
1: privileged

13.8.2 GPDMA configuration lock register (GPDMA_RCFGLOCKR)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be written by a software agent with privileged attributes in order to individually lock, for example at boot time, the privileged attributes of any GPDMA channel/resource (to lock the setting of GPDMA_CxPRIVCFGR for any channel x, for example at boot time).

A read access may be privileged or unprivileged.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **LOCKx**: lock the configuration of GPDMA_PRIVCFGR.PRIVx until a global GPDMA reset (x = 15 to 0)
This bit is cleared after reset and, once set, it cannot be reset until a global GPDMA reset.
0: privilege configuration of the channel x is writable.
1: privilege configuration of the channel x is not writable.

13.8.3 GPDMA masked interrupt status register (GPDMA_MISR)

Address offset: 0x0C

Reset value: 0x0000 0000

This is a read register.

This register contains the masked interrupt status bit MISx for each channel x. It is a logical OR of all the GPDMA_CxSR flags, each source flag being enabled by the corresponding GPDMA_CxCR interrupt enable bit.

Every bit is deasserted by hardware when writing 1 to the corresponding GPDMA_CxFCR flag clear bit.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel GPDMA_PRIVCFGR.PRIVx. A privileged software can read the full interrupt status. An unprivileged software is restricted to read the status of unprivileged channels, other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MISx**: masked interrupt status of channel x (x = 15 to 0)
0: no interrupt occurred on channel x
1: an interrupt occurred on channel x

13.8.4 GPDMA channel x linked-list base address register (GPDMA_CxLBAR)

Address offset: 0x50 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register must be written by a privileged software. It is either privileged readable or not, depending on the privileged state of the channel x GPDMA_PRIVCFGR.PRIVx.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This channel-based register is the linked-list base address of the memory region, for a given channel x, from which the LLIs describing the programmed sequence of the GPDMA transfers, are conditionally and automatically updated.

This 64-Kbyte aligned channel x linked-list base address is offset by the 16-bit GPDMA_CxLLR register that defines the word-aligned address offset for each LLI.

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

Bits 31:16 **LBA[31:16]**: linked-list base address of GPDMA channel x

Bits 15:0 Reserved, must be kept at reset value.

13.8.5 GPDMA channel x flag clear register (GPDMA_CxFCR)

Address offset: 0x5C+ 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This is a write register privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	w	w	w	w	w	w	w								

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag clear
0: no effect
1: corresponding TOF flag cleared

Bit 13 **SUSPF**: completed suspension flag clear
0: no effect
1: corresponding SUSPF flag cleared

Bit 12 **USEF**: user setting error flag clear
0: no effect
1: corresponding USEF flag cleared

Bit 11 **ULEF**: update link transfer error flag clear
0: no effect
1: corresponding ULEF flag cleared

Bit 10 **DTEF**: data transfer error flag clear
0: no effect
1: corresponding DTEF flag cleared

Bit 9 **HTF**: half-transfer flag clear
0: no effect
1: corresponding HTF flag cleared

Bit 8 **TCF**: transfer complete flag clear
0: no effect
1: corresponding TCF flag cleared

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

13.8.6 GPDMA channel x status register (GPDMA_CxSR)

Address offset: 0x60 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0001

This is a read register, reporting the channel status.

This register is privileged or unprivileged, depending on the privileged state of the channel (GPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	DRAINING F	IDLEF
	r	r	r	r	r	r	r							r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **FIFOL[7:0]**: monitored FIFO level (in bytes)
Number of available bytes in the FIFO.

Note: After having suspended an active transfer, the user may need to read FIFOL[7:0], additionally to GPDMA_CxBR1.BDNT[15:0] and GPDMA_CxBR1.BRC[10:0], to know how many data have been transferred to the destination. Before reading, the user may wait for the transfer to be suspended (GPDMA_CxSR.SUSPF = 1).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag
0: no trigger overrun event
1: a trigger overrun event occurred

Bit 13 **SUSPF**: completed suspension flag
0: no completed suspension event
1: a completed suspension event occurred

Bit 12 **USEF**: user setting error flag
0: no user setting error event
1: a user setting error event occurred

Bit 11 **ULEF**: update link transfer error flag
0: no update link transfer error event
1: a master bus error event occurred while updating a linked-list register from memory

Bit 10 **DTEF**: data transfer error flag
0: no data transfer error event
1: a master bus error event occurred on a data transfer

Bit 9 **HTF**: half-transfer flag
0: no half-transfer event
1: a half-transfer event occurred
A half-transfer event is either a half-block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode (GPDMA_CxTR2.TCEM[1:0]).
A half-block transfer occurs when half of the bytes of the source block size (rounded up integer of GPDMA_CxBR1.BNDT[15:0]/2) has been transferred to the destination.
A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 2)) has been transferred to the destination.

Bit 8 **TCF**: transfer complete flag
0: no transfer complete event
1: a transfer complete event occurred
A transfer complete event is either a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode (GPDMA_CxTR2.TCEM[1:0]).

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

Bit 1 **DRAININGF**: FIFO draining flag
0: No received order to drain the FIFO or channel FIFO draining completed
1: Ongoing channel FIFO draining and channel termination
This flag is asserted after that the order to drain the FIFO is received, and meanwhile that the channel is enabled and draining the FIFO and before the channel is terminated with a drained FIFO.
This flag is deasserted by hardware when the enabled channel has been ordered to be terminated with FIFO draining, and the hardware effectively terminates.

Bit 0 **IDLEF**: idle flag
0: channel not in idle state
1: channel in idle state
This idle flag is deasserted by hardware when the channel is enabled (GPDMA_CxCR.EN = 1) with a valid channel configuration (no USEF to be immediately reported).
This idle flag is asserted after hard reset or by hardware when the channel is back in idle state (in suspended or disabled state).

13.8.7 GPDMA channel x control register (GPDMA_CxCR)

Address offset: 0x64 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

This register is used to control a channel (activate, suspend, abort or disable it).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	LAP	LSM
								rw	rw					rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	DRAIN FIFO	SUSP	RESET	EN
	rw	rw	rw	rw	rw	rw	rw					w	rw	w	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **PRIO[1:0]:** priority level of the channel x GPDMA transfer versus others
00: low priority, low weight
01: low priority, mid weight
10: low priority, high weight
11: high priority
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 21:18 Reserved, must be kept at reset value.

Bit 17 **LAP**: linked-list allocated port
This bit is used to allocate the master port for the update of the GPDMA linked-list registers from the memory.
0: port 0 (AHB) allocated
1: port 1 (AHB) allocated
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 16 **LSM**: Link step mode
0: channel executed for the full linked-list and completed at the end of the last LLI (GPDMA_CxLLR = 0). The 16 low-significant bits of the link address are null (LA[15:0] = 0) and all the update bits are null (UT1 = UB1 = UT2 = USA = UDA = ULL = 0 and UT3 = UB2 = 0). Then GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0.
1: channel executed **once** for the current LLI
First the (possible 1D/repeated) block transfer is executed as defined by the current internal register file until GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Secondly the next linked-list data structure is conditionally uploaded from memory as defined by GPDMA_CxLLR. Then channel execution is completed.
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOIE**: trigger overrun interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 13 **SUSPIE**: completed suspension interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 12 **USEIE**: user setting error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 11 **ULEIE**: update link transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 10 **DTEIE**: data transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 9 **HTIE**: half-transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 8 **TCIE**: transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **DRAINFIFO:** empty FIFO
This bit is write only. Writing 0 has no impact.
Writing 1 to RESET in this register causes the hardware to deassert this DRAINFIFO bit, whatever is written into this DRAINFIFO. Else:
Software must write 1 in order to drain the channel FIFO on a suspended channel.
If the software concurrently writes 1 to DRAINFIFO and 1 to SUSP and 0 to reset, the hardware suspends the channel and drains the FIFO by popping out the residual FIFO bytes to the destination (see the programming sequence in *Figure 105*).
0: Write: no action
1: Write: drain FIFO after having suspended the channel

Bit 2 **SUSP:** suspend
Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 2. Else:
Software must write 1 in order to suspend an active channel (channel with an ongoing GPDMA transfer over its master ports).
The software must write 0 in order to resume a suspended channel, following the programming sequence detailed in *Figure 103*.
0: write: resume channel, read: channel not suspended
1: write: suspend channel, read: channel suspended.

Bit 1 **RESET:** reset

This bit is write only. Writing 0 has no impact. Writing 1 implies the reset of the following: the FIFO, the channel internal state, SUSP, DRAINFIFO and EN bits (whatever is written receptively in bit 3 and bit 2 and bit 0).

The reset is effective when the channel is in steady state, meaning one of the following:

- active channel in suspended state (GPDMA_CxSR.SUSPF = 1, GPDMA_CxSR.IDLEF = GPDMA_CxCR.EN = 1 and GPDMA_CxSR.DRAININGF = 0)
- channel in disabled state (GPDMA_CxSR.IDLEF = 1 and GPDMA_CxCR.EN = 0).

After writing a RESET, to continue using this channel, the user must explicitly reconfigure the channel including the hardware-modified configuration registers (GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR) before enabling again the channel (see the programming sequence in *Figure 104*).

0: no channel reset

1: channel reset

Bit 0 **EN:** enable

Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 0. Else:

this bit is deasserted by hardware when there is a transfer error (master bus error or user setting error) or when there is a channel transfer complete (channel ready to be configured, for example if LSM = 1 at the end of a single execution of the LLI).

Else, this bit can be asserted by software.

Writing 0 into this EN bit is ignored.

0: write: ignored, read: channel disabled

1: write: enable channel, read: channel enabled

13.8.8 GPDMA channel x transfer register 1 (GPDMA_CxTR1)

Address offset: 0x90 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of the channel x in GPDMA_PRIVCFGR.PRIVx.

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed. Then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UT1 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DAP	Res.	Res.	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2[1:0]	
	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2[1:0]	
	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DAP**: destination allocated port

This bit is used to allocate the master port for the destination transfer

0: port 0 (AHB) allocated

1: port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **DHX**: destination half-word exchange

If the destination data size is shorter than a word, this bit is ignored.

If the destination data size is a word:

0: no halfword-based exchanged within word

1: the two consecutive (post PAM) half-words are exchanged in each destination word.

Bit 26 **DBX**: destination byte exchange

If the destination data size is a byte, this bit is ignored.

If the destination data size is not a byte:

0:no byte-based exchange within half-word

1: the two consecutive (post PAM) bytes are exchanged in each destination half-word.

Bits 25:20 **DBL_1[5:0]**: destination burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If DBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width DDW_LOG2[1:0].

Note: If a burst transfer crossed a 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 19 **DINC**: destination incrementing burst

0: fixed burst

1: contiguously incremented burst

The destination address, pointed by GPDMA_CxDAR, is kept constant after a burst beat/single transfer, or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DDW_LOG2[1:0]**: binary logarithm of the destination data width of a burst, in bytes

00: byte

01: half-word (2 bytes)

10: word (4 bytes)

11: user setting error reported and no transfer issued

Note: A destination burst transfer must have an aligned address with its data width (start address GPDMA_CxDAR[2:0] and if present address offset GPDMA_CxTR3.DAO[2:0], versus DDW_LOG2[1:0]). Otherwise a user setting error is reported and no transfer is issued.

Setting a 8-byte data width causes a user setting error to be reported and none transfer is issued.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SAP**: source allocated port

This bit is used to allocate the master port for the source transfer

0: port 0 (AHB) allocated

1: port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 13 **SBX**: source byte exchange within the unaligned half-word of each source word

If the source data width is shorter than a word, this bit is ignored.

If the source data width is a word:

0:no byte-based exchange within the unaligned half-word of each source word

1: the two consecutive bytes within the unaligned half-word of each source word are exchanged.

Bits 12:11 **PAM[1:0]**: padding/alignment mode

If DDW_LOG2[1:0] = SDW_LOG2[1:0]: if the data width of a burst destination transfer is equal to the data width of a burst source transfer, these bits are ignored.

Else, in the following enumerated values, the condition PAM_1 is when destination data width is higher than source data width, and the condition PAM_2 is when source data width is higher than destination data width.

Condition: PAM_1

00: source data is transferred as right aligned, padded with 0s up to the destination data width

01: source data is transferred as right aligned, sign extended up to the destination data width

10-11: successive source data are FIFO queued and packed at the destination data width, in a left (LSB) to right (MSB) order (named little endian), before a destination transfer

Condition: PAM_2

00: source data is transferred as right aligned, left-truncated down to the destination data width

01: source data is transferred as left-aligned, right-truncated down to the destination data width

10-11: source data is FIFO queued and unpacked at the destination data width, to be transferred in a left (LSB) to right (MSB) order (named little endian) to the destination

Note: If the transfer from the source peripheral is configured with peripheral flow-control mode (SWREQ = 0 and PFREQ = 1 and DREQ = 0), and if the destination data width > the source data width, packing is not supported.

Bit 10 Reserved, must be kept at reset value.

Bits 9:4 **SBL_1[5:0]**: source burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If SBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width SDW_LOG2[1:0].

Note: If a burst transfer crossed a 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 3 **SINC**: source incrementing burst
0: fixed burst
1: contiguously incremented burst
The source address, pointed by GPDMA_CxSAR, is kept constant after a burst beat/single transfer or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SDW_LOG2[1:0]**: binary logarithm of the source data width of a burst in bytes
00: byte
01: half-word (2 bytes)
10: word (4 bytes)
11: user setting error reported and no transfer issued
A source burst transfer must have an aligned address with its data width (start address GPDMA_CxSAR[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.
Setting a 8-byte data width causes a user setting error to be reported and no transfer is issued.

13.8.9 GPDMA channel x transfer register 2 (GPDMA_CxTR2)

Address offset: 0x94 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x. This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (the hardware deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory, if GPDMA_CxLLR.UT2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	Res.	TRIGSEL[5:0]					
rw	rw					rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGM[1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	REQSEL[7:0]							
rw	rw		rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **TCEM[1:0]**: transfer complete event mode

These bits define the transfer granularity for the transfer complete and half-transfer complete events generation.

00: at block level (when GPDMA_CxBR1.BNDT[15:0] = 0): the complete (and the half) transfer event is generated at the (respectively half of the) end of a block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half-transfer event is generated.

01: channel x (x = 0 to 11), same as 00, channel x (x =12 to 15), at 2D/repeated block level (when GPDMA_CxBR1.BRC[10:0] = 0 and GPDMA_CxBR1.BNDT[15:0] = 0). The complete (and the half) transfer event is generated at the end (respectively half of the end) of the 2D/repeated block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half-transfer event is generated.

10: at LLI level: the complete transfer event is generated at the end of the LLI transfer, including the update of the LLI if any. The half-transfer event is generated at the half of the LLI data transfer The LLI data transfer is a block transfer or a 2D/repeated block transfer for channel x (x =12 to 15), if any data transfer.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then the half-transfer event is not generated, and the transfer complete event is generated when is completed the loading of the LLI_1.

11: at channel level: the complete transfer event is generated at the end of the last LLI transfer. The half-transfer event is generated at the half of the data transfer of the last LLI. The last LLI updates the link address GPDMA_CxLLR.LA[15:2] to zero and clears all the GPDMA_CxLLR update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UT3 and UB2). If the channel transfer is continuous/infinite, no event is generated.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **TRIGPOL[1:0]**: trigger event polarity

These bits define the polarity of the selected trigger event input defined by TRIGSEL[5:0].

00: no trigger (masked trigger event)

01: trigger on the rising edge

10: trigger on the falling edge

11: same as 00

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **TRIGSEL[5:0]**: trigger event input selection

These bits select the trigger event input of the GPDMA transfer (as per *Section 13.3.6*), with an active trigger event if TRIGPOL[1:0] ≠ 00.

Bits 15:14 **TRIGM[1:0]**: trigger mode

These bits define the transfer granularity for its conditioning by the trigger.

If the channel x is enabled (GPDMA_CxCR.EN asserted) with TRIGPOL[1:0] = 00 or 11, these TRIGM[1:0] bits are ignored.

Else, a GPDMA transfer is conditioned by one trigger hit:

00: at block level: the first burst read of each block transfer is conditioned by one hit trigger (channel x (x = 12 to 15): same, each block also if a 2D/repeated block is configured with GPDMA_CxBR1.BRC[10:0] ≠ 0).

01: channel x (x = 0 to 11): same as 00; channel x (x = 12 to 15): at 2D/repeated block level: the first burst read of the 2D/repeated block transfer is conditioned by one hit trigger.

10: at link level: a LLI link transfer is conditioned by one hit trigger. LLI data transfer is not conditioned.

11: at programmed burst level: If SWREQ = 1, each programmed burst read is conditioned by one hit trigger. If SWREQ = 0, each programmed burst that is requested by the selected peripheral, is conditioned by one hit trigger:

– If the peripheral is the source (DREQ = 0) of the LLI data transfer, each programmed burst read is conditioned.

– If the peripheral is the destination (DREQ = 1) of the LLI data transfer, each programmed burst write is conditioned. The first memory burst read of a (possibly 2D/repeated) block, also named as the first ready FIFO-based source burst, is gated by the occurrence of both the hardware request and one trigger hit.

As shown in *Figure 120: Trigger hit, memorization, and overrun waveform*, the GPDMA monitoring of a trigger for channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (TRIGPOL[1:0] = 01 or respectively TRIGPOL[1:0] = 10). The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer; and if a new trigger is detected then, this hit is internally memorized to grant the next transfer, as long as the defined rising or falling edge is not modified, and the TRIGSEL[5:0] is not modified, and the channel is enabled. After a first new trigger hit_{n+1} is memorized, if another second trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (GPDMA_CxSR.TOF = 1), and an interrupt is generated if enabled (GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Transferring a next LLI_{n+1} that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the memorized hit of the formerly defined LLI_n trigger.

Note: When the source block size is not a multiple of the source burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, if TRIGM[1:0] = 11 and (SWREQ = 1 or (SWREQ = 0 and DREQ = 0)), the shortened burst transfer (by singles or/and by bursts of lower length) is conditioned once by the trigger.

When the programmed destination burst is internally shortened by singles or/and by bursts of lower length (versus FIFO size, versus block size, 1-Kbyte boundary address crossing): if the trigger is conditioning the programmed destination burst (if TRIGM[1:0] = 11 and SWREQ = 0 and DREQ = 1), this shortened destination burst transfer is conditioned once by the trigger.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PFREQ**: Hardware request in peripheral flow control mode

Important: If a given channel x is not implemented with this feature, this bit is reserved and PFREQ is not present (see Section 13.3.1 for the list of the implemented channels with this feature).

If the channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in GPDMA control mode. The GPDMA is programmed with GPDMA_CxTR1.BNDT[15:0] and this is internally used by the hardware for the block transfer completion.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in peripheral control mode. The GPDMA block transfer can be early completed by the peripheral itself (see *Section 13.3.5* for more details).

Note: In peripheral flow control mode, there are the following restrictions:

- no 2D/repeated block support (GPDMA_CxBR1.BRC[10:0] must be set to 0)

- the peripheral must be set as the source of the transfer (DREQ = 0).

- data packing to a wider destination width is not supported (if destination width > source data width, GPDMA_CxTR1.PAM[1] must be set to 0).

Bit 11 **BREQ**: Block hardware request

If the channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a burst level.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a block level (see *Section 13.3.3*).

Bit 10 **DREQ**: destination hardware request

This bit is ignored if channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else:

0: selected hardware request driven by a source peripheral (request signal taken into account by the GPDMA transfer scheduler over the source/read port)

1: selected hardware request driven by a destination peripheral (request signal taken into account by the GPDMA transfer scheduler over the destination/write port)

Note: If the channel x is activated (GPDMA_CxCR.EN is asserted) with SWREQ = 0 and PFREQ = 1 (peripheral hardware request with peripheral flow-control mode), any software assertion to this DREQ bit is ignored: in peripheral flow-control mode, only a peripheral-to-memory transfer is supported.

Bit 9 **SWREQ**: software request

This bit is internally taken into account when GPDMA_CxCR.EN is asserted.

0: no software request. The selected hardware request REQSEL[7:0] is taken into account.

1: software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[7:0] is ignored.

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **REQSEL[7:0]**: GPDMA hardware request selection

These bits are ignored if channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1(software request for a memory-to-memory transfer). Else, the selected hardware request is internally taken into account as per *Section 13.3.3*.

Caution: The user must not assign a same input hardware request (same REQSEL[7:0] value) to different active GPDMA channels (GPDMA_CxCR.EN = 1 and GPDMA_CxTR2.SWREQ = 0 for these channels). GPDMA is not intended to hardware support the case of simultaneous enabled channels incorrectly configured with a same hardware peripheral request signal, and there is no user setting error reporting.

13.8.10 GPDMA channel x block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the field BNDT[15:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for BNDT[15:0] after each execution of this final LLI
- If GPDMA_CxLLR = 0, this register and BNDT[15:0] are kept as null, channel x is completed.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if GPDMA_CxLLR.UB1 = 1, this field is updated with the LLI in the memory.
- if GPDMA_CxLLR.UB1 = 0 and if there is at least one non null Uxx update bit, this field is internally restored to the programmed value.
- if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] = 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
- if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

13.8.11 GPDMA channel x alternate block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0] after each execution of this final LLI
- If GPDMA_CxLLR = 0, BNDT[15:0] and BRC[10:0] are kept as null, channel x is completed.

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

Bit 31 **BRDDEC**: Block repeat destination address decrement

0: at the end of a block transfer, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxBR2.BRDAO to the current GPDMA_CxDAR value (current destination address)

1: at the end of a block transfer, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRDAO from the current GPDMA_CxDAR value (current destination address)

Note: On top of this increment/decrement (depending on BRDDEC), GPDMA_CxDAR is in the same time also updated by the increment/decrement (depending on DDEC) of the GPDMA_CxTR3.DAO value, as it is usually done at the end of each programmed burst transfer.

Bit 30 **BRSDEC**: Block repeat source address decrement

0: at the end of a block transfer, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxBR2.BRSAO to the current GPDMA_CxSAR value (current source address)

1: at the end of a block transfer, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRSAO from the current GPDMA_CxSAR value (current source address)

Note: On top of this increment/decrement (depending on BRSDEC), GPDMA_CxSAR is in the same time also updated by the increment/decrement (depending on SDEC) of the GPDMA_CxTR3.SAO value, as it is done after any programmed burst transfer.

Bit 29 **DDEC**: destination address decrement

0: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

1: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

Bit 28 **SDEC**: source address decrement

0: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

1: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **BRC[10:0]**: Block repeat counter

This field contains the number of repetitions of the current block (0 to 2047).

When the channel is enabled, this field becomes read-only. After decrements, this field indicates the remaining number of blocks, excluding the current one. This counter is hardware decremented for each completed block transfer.

Once the last block transfer is completed (BRC[10:0] = BNDT[15:0] = 0):

– If GPDMA_CxLLR.UB1 = 1, all GPDMA_CxBR1 fields are updated by the next LLI in the memory.
– If GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.
– if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
– if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI and data transfer.

Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if GPDMA_CxLLR.UB1 = 1, this field is updated with the LLI in the memory.
- if GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.
- if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
- if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

13.8.12 GPDMA channel x source address register (GPDMA_CxSAR)

Address offset: 0x9C + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the source start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer from the source.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.USA = 1.

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

Bits 31:0 **SA[31:0]**: source address

This field is the pointer to the address from which the next data is read.

During the channel activity, depending on the source addressing mode (GPDMA_CxTR1.SINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.SDW_LOG2[1:0]) after each burst source data, reflecting the next address from which data is read.

During the channel activity, this address is updated after each completed source burst, consequently to:

– the programmed source burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.SINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.SBL_1[5:0] and GPDMA_CxTR1.SDW_LOG2[1:0]

– the additional source incremented/decremented offset value as programmed by GPDMA_CxBR1.SDEC and GPDMA_CxTR3.SAO[12:0].

– once/if completed source block transfer, for a channel x with 2D addressing capability (x = 12 to 15). additional block repeat source incremented/decremented offset value as programmed by GPDMA_CxBR1.BRSDEC and GPDMA_CxBR2.BRSAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.USA = 1.

Note: A source address must be aligned with the programmed data width of a source burst (SA[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

When the source block size is not a multiple of the source burst size and is a multiple of the source data width, the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.

13.8.13 GPDMA channel x destination address register (GPDMA_CxDAR)

Address offset: 0xA0 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the destination start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer to the destination.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UDA = 1.

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

Bits 31:0 **DA[31:0]**: destination address

This field is the pointer to the address from which the next data is written.

During the channel activity, depending on the destination addressing mode (GPDMA_CxTR1.DINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.DDW_LOG2[1:0]) after each burst destination data, reflecting the next address from which data is written.

During the channel activity, this address is updated after each completed destination burst, consequently to:

– the programmed destination burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.DINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.DBL_1[5:0] and GPDMA_CxTR1.DDW_LOG2[1:0]

– the additional destination incremented/decremented offset value as programmed by GPDMA_CxBR1.DDEC and GPDMA_CxTR3.DAO[12:0].

– once/if completed destination block transfer, for a channel x with 2D addressing capability (x = 12 to 15), the additional block repeat destination incremented/decremented offset value as programmed by GPDMA_CxBR1.BRDDEC and GPDMA_CxBR2.BRDAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by the GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.UDA = 1.

Note: A destination address must be aligned with the programmed data width of a destination burst (DA[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

13.8.14 GPDMA channel x transfer register 3 (GPDMA_CxTR3)

Address offset: 0xA4 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UT3 = 1.

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **DAO[12:0]**: destination address offset increment

The destination address, pointed by GPDMA_CxDAR, is incremented or decremented (depending on GPDMA_CxBR1.DDEC) by this offset DAO[12:0] for each programmed destination burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.DINC = 1).

Note: A destination address offset must be aligned with the programmed data width of a destination burst (DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **SAO[12:0]**: source address offset increment

The source address, pointed by GPDMA_CxSAR, is incremented or decremented (depending on GPDMA_CxBR1.SDEC) by this offset SAO[12:0] for each programmed source burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.SINC = 1).

Note: A source address offset must be aligned with the programmed data width of a source burst (SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

When the source block size is not a multiple of the destination burst size, and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.

13.8.15 GPDMA channel x block register 2 (GPDMA_CxBR2)

Address offset: 0xA8 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a 2D/repeated block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UB2 = 1.

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

Bits 31:16 **BRDAO[15:0]**: Block repeated destination address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRDDEC) the current destination address (GPDMA_CxDAR) at the end of a block transfer.

Note: A block repeated destination address offset must be aligned with the programmed data width of a destination burst (BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued. BRDAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).

Bits 15:0 **BRSAO[15:0]**: Block repeated source address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRSDEC) the current source address (GPDMA_CxSAR) at the end of a block transfer.

Note: A block repeated source address offset must be aligned with the programmed data width of a source burst (BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued. BRSAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).

13.8.16 GPDMA channel x linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw											rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update GPDMA_CxTR1 from memory

This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.
0: no GPDMA_CxTR1 update
1: GPDMA_CxTR1 update

Bit 30 **UT2**: Update GPDMA_CxTR2 from memory

This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.
0: no GPDMA_CxTR2 update
1: GPDMA_CxTR2 update

Bit 29 **UB1**: Update GPDMA_CxBR1 from memory

This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed.
GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.
0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)
1: GPDMA_CxBR1 update

Bit 28 **USA**: update GPDMA_CxSAR from memory

This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.
0: no GPDMA_CxSAR update
1: GPDMA_CxSAR update

Bit 27 **UDA**: Update GPDMA_CxDAR register from memory

This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.

0: no GPDMA_CxDAR update

1: GPDMA_CxDAR update

Bits 26:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update GPDMA_CxLLR register from memory

This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.

0: no GPDMA_CxLLR update

1: GPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure

If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.

Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR).

Caution: The user must program this pointer not to exceed the 64-Kbyte addressable space defined by the link base address register GPDMA_CxLBAR. The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

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

13.8.17 GPDMA channel x alternate linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw	rw	rw									rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update GPDMA_CxTR1 from memory

This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.

0: no GPDMA_CxTR1 update

1: GPDMA_CxTR1 update

Bit 30 **UT2**: Update GPDMA_CxTR2 from memory

This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.

0: no GPDMA_CxTR2 update

1: GPDMA_CxTR2 update

Bit 29 **UB1**: Update GPDMA_CxBR1 from memory

This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed.
GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.

0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)

1: GPDMA_CxBR1 update

Bit 28 **USA**: update GPDMA_CxSAR from memory

This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.

0: no GPDMA_CxSAR update

1: GPDMA_CxSAR update

Bit 27 **UDA**: Update GPDMA_CxDAR register from memory
This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.
0:no GPDMA_CxDAR update
1: GPDMA_CxDAR update

Bit 26 **UT3**: Update GPDMA_CxTR3 from memory
This bit controls the update of GPDMA_CxTR3 from the memory during the link transfer.
0: no GPDMA_CxTR3 update
1: GPDMA_CxTR3 update

Bit 25 **UB2**: Update GPDMA_CxBR2 from memory
This bit controls the update of GPDMA_CxBR2 from the memory during the link transfer.
0: no GPDMA_CxBR2 update
1: GPDMA_CxBR2 update

Bits 24:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update GPDMA_CxLLR register from memory
This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.
0: no GPDMA_CxLLR update
1: GPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.
Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR).

Caution: The user must program this pointer not to exceed the 64-Kbyte addressable space defined by the link base address register GPDMA_CxLBAR. The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

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

13.8.18 GPDMA register map

Table 104. GPDMA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x04	GPDMA_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPDMA_ RCFGLOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 104. GPDMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	GPDMA_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	GPDMA_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14 - 0x4C	Reserved	Reserved																															
0x50+0x80 * x (x = 0 to 15)	GPDMA_ CxLBAR	LBA[31:16]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x5C+0x80 * x (x = 0 to 15)	GPDMA_CxFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0	0	0	0	0	0	0								
0x60+0x80 * x (x = 0 to 15)	GPDMA_CxSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[7:0]								Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	DRAININGF	IDLEF
	Reset value									0	0	0	0	0	0	0	0		0	0	0	0	0	0	0							0	1
0x64+0x80 * x (x = 0 to 15)	GPDMA_CxCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO [1:0]		Res.	Res.	Res.	Res.	LAP	LSM	Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	DRAINFIFO	SUSP	RESET	EN
	Reset value									0	0					0	0		0	0	0	0	0	0	0					0	0	0	0
0x90+0x80 * x (x = 0 to 15)	GPDMA_CxTR1	Res.	DAP	Res.	Res.	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2 [1:0]		Res.	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2 [1:0]	
	Reset value		0			0	0	0	0	0	0	0	0	0		0	0		0	0	0	0		0	0	0	0	0	0	0		0	0
0x94+0x80 * x (x = 0 to 15)	GPDMA_CxTR2	TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL [1:0]		Res.	Res.	TRIGSEL[5:0]						TRIGM [1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	REQSEL[7:0]							
	Reset value	0	0					0	0			0	0	0	0	0	0	0	0		0	0	0	0		0	0	0	0	0	0	0	0
0x98+0x80 * x (x = 0 to 11)	GPDMA_CxBR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BNDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98+0x80 * x (x = 12 to 15)	GPDMA_CxBR1	BRDDEC	BRSDEC	DDEC	SDEC	Res.	BRC[10:0]											BNDT[15:0]															
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9C+0x80 * x (x = 0 to 15)	GPDMA_CxSAR	SA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA0+0x80 * x (x = 0 to 15)	GPDMA_CxDAR	DA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4+0x80 * x (x = 12 to 15)	GPDMA_CxTR3	Res.	Res.	Res.	DAO[12:0]													Res.	Res.	Res.	SAO[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0xA8+0x80 * x (x = 12 to 15)	GPDMA_CxBR2	BRDAO[15:0]																BRSAO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xCC+0x80 * x (x = 0 to 11)	GPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0xCC+0x80 * x (x = 12 to 15)	GPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

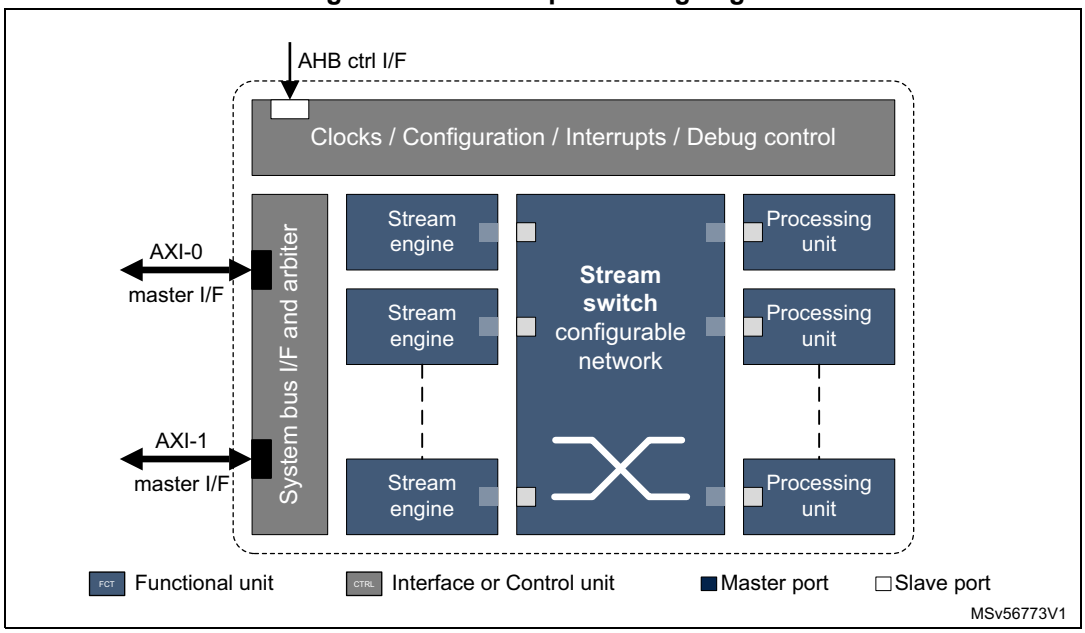
Refer to *Section 2.3: Memory organization* for the register boundary addresses.

14 Neural-ART accelerator™ (NPU)

14.1 NPU introduction

Neural-ART accelerator is a branded family of design-time parametric and runtime reconfigurable neural processing unit (NPU) cores. These cores are specifically designed to accelerate (in area-constrained embedded and IoT devices) the inference execution of a wide range of quantized convolutional neural network (CNN) models.

Figure 123. Stream processing engine



Stream processing

The ST stream processing technology allows the creation of on-demand performance- and area-tailored instances to address artificial intelligence (AI) applications, from the lowest to the highest end class.

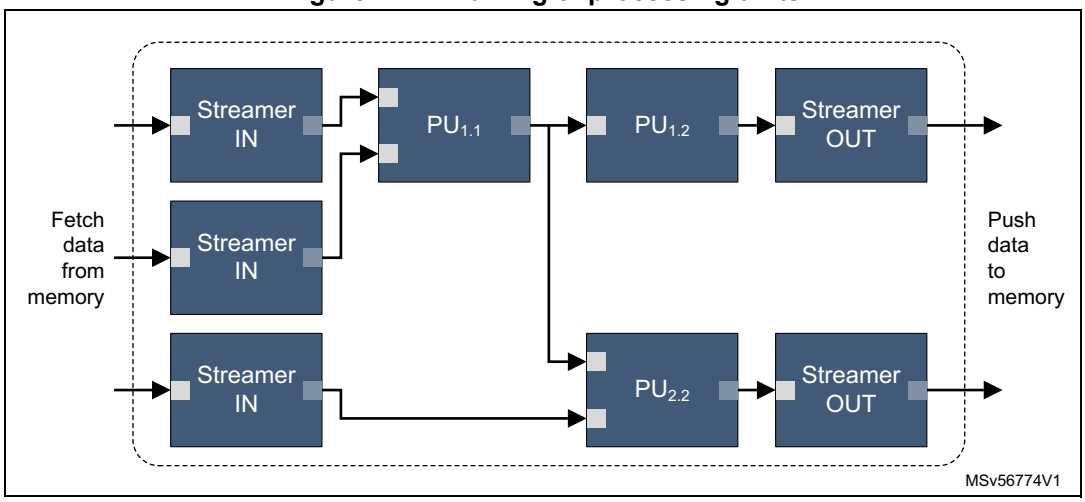
The ST stream processing technology is deemed far more flexible than a hardware data path, and more power-efficient than an on-chip memory bus. The core is a configurable data flow fabric, also known as stream switch, which can be programmed at runtime to create arbitrary virtual chains of processing units (PUs), implementing the sequence of operations to execute. Multiple chains can be active at the same time, provided there are enough available hardware resources.

In the virtual chains, data flow unidirectionally from one PU to the other, hence a processing chain cannot be faster than the slowest element in the chain. A back pressure mechanism controls the data flow in a virtual chain, and data streams can be easily forked or joined to feed multiple destinations, enabling multicasting, context switching, and filtering.

Chaining as many accelerators as possible in a single virtual chain reduces the number of useless and power consuming transfers from/to memory, thus reducing the overall memory traffic and the power consumption.

The resulting IP communicates with the rest of the system through stream engine units. These are smart half-DMAs that can read, and write the data to/from external memory to transfer from/to the IP internal buffers. Although not primarily intended for that, memory-to-memory transfers are possible, connecting two stream engines back to back.

Figure 124. Chaining of processing units



Epoch definition

The execution of CNN models is organized into indivisible processing sequences called epochs. A hardware epoch is defined by the sequence that starts with the configuration of the NPU by the host CPU, and ends with an interrupt sent back to the CPU. The host CPU can then trigger another hardware epoch or execute some software computing for the layers not accelerated by the hardware.

14.2 NPU implementation

The Neural-ART 11 NPU uses a 1CA configuration with the functional units listed in *Table 105*.

Table 105. Neural-ART 11 NPU configuration

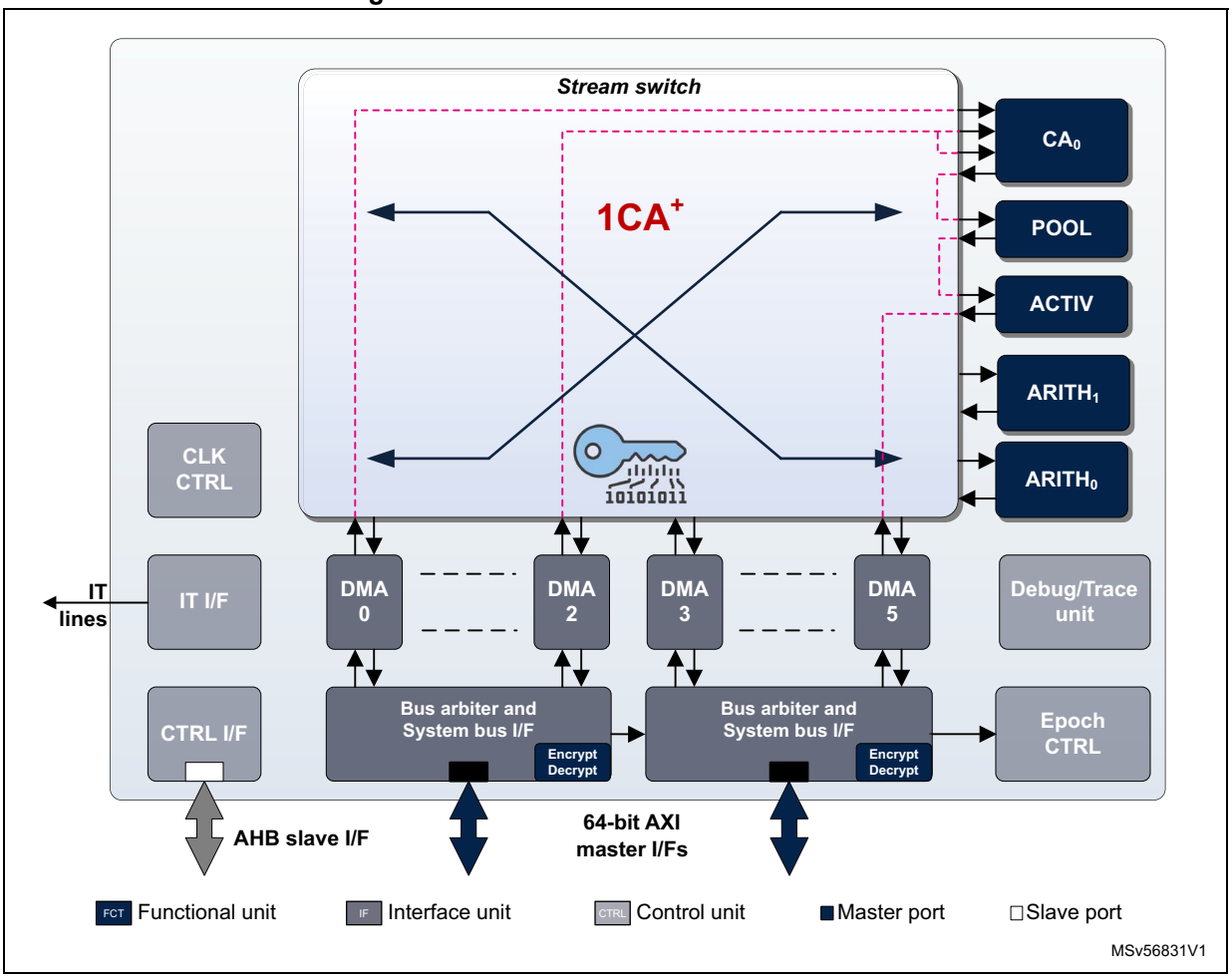
Count	Unit	Description
1	STRSWITCH	Central data stream switch
1	CONVACC	*Convolutional accelerator*
1	POOL	*Pooling unit*
1	ACTIV	*Activation unit*
2	ARITH	*Arithmetic unit* for scalar arithmetic operators
6	STRENG	*Stream engines*, with embedded cipher
2	BUSIF	64-bit AXI interface
1	EPOCH	*Epoch controller*
1	DEBUG	Debug controller
-	SERVICES	Configuration network, Clock and reset manager, Interrupt controller

14.3 NPU functional description

14.3.1 NPU block diagram

The internal organization of the NPU is shown in *Figure 125*.

Figure 125. Neural-ART 11 NPU accelerator



When the NPU is configured and started, it autonomously fetches data from the external memory to feed its internal dynamically interconnected processing units. Similarly, the result of the operation sequence is flushed to the external memory.

Inputs/sources can be the activations/features, but also parameters, such as weights and biases constant data hosted in a memory-mapped nonvolatile memory. When the data-stream processing pipes are configured and started, the settings cannot be modified until the end-of-transfer event notifications of all DMA-outs and sinks.

14.3.2 NPU pins and internal signals

The NPU has no external connections.

Table 106. NPU internal signals

Signal name	Signal type	Comment
npu_clk	Input	Main clock
npu_rstn	Input	Reset
npu_host_int[3:0]	Output	Host interrupts
dbg_freeze_async	Input	Freeze in debug
ext_triggers_async[3:0]	Input	External synchronization triggers
trig_sig[7:0]	Output	Debug and trace triggers to CTI

14.3.3 Configuration network

Each of the functional units, whether accelerator or interface, has a configuration space made of a collection of configuration registers. These registers are accessible through the AHB interface. Their detailed description is out of the scope of the present document.

The NPU embeds a configuration network that provides host processor access to the configuration space of the different units in the Neural-ART core. This configuration network connects to the system backbone through a 32-bit AHB5-lite port. It supports a single 32-bit wide read and write transaction, with no pipelining nor outstanding transactions.

14.3.4 Clock/reset manager

The purpose of this unit is twofold:

1. Clock control

   The NPU uses a single functional clock provided by the RCC. This input clock is fed to the clock control unit from where it is gated and fed to the different units of the IP.

   The clock unit generates clocks not gated, and clocks gated by default that need to be enabled after reset.

   Clock gating is controlled by a configuration register bank, accessible via the configuration network, and programmed by the control code generated by the NPU compiler tool chain.

2. Reset control

   The NPU uses a single functional reset provided by the RCC. This reset signal is synchronized inside the IP and fed to the different functional units.

   Setting the clear bit in the control configuration register triggers a global NPU reset.

14.3.5 Stream link

A stream link is a unidirectional interconnection to transport data streams from accelerator units, stream engines, and interfaces to the stream switch and vice versa.

The stream consists of data, messages, or commands, and additional signaling information.

- Data are transported over three 8-bit wide data channels that are either handled individually or can be interpreted as a single 24-bit data value.
- Transported data can use different formats. A raster scan data stream allows the automatic detection of the geometrical dimensions of the received feature data tensor (width and height). A raw data format requires to program additional parameters to handle correctly the received data stream.
- Signaling information is used to identify valid information on the stream link. It indicates the type and format of information transported, and whether this is the first or the last data transmitted.

14.3.6 Stream switch

This switch dynamically connects output stream link ports to input stream link ports, depending on the configuration data written to the configuration registers.

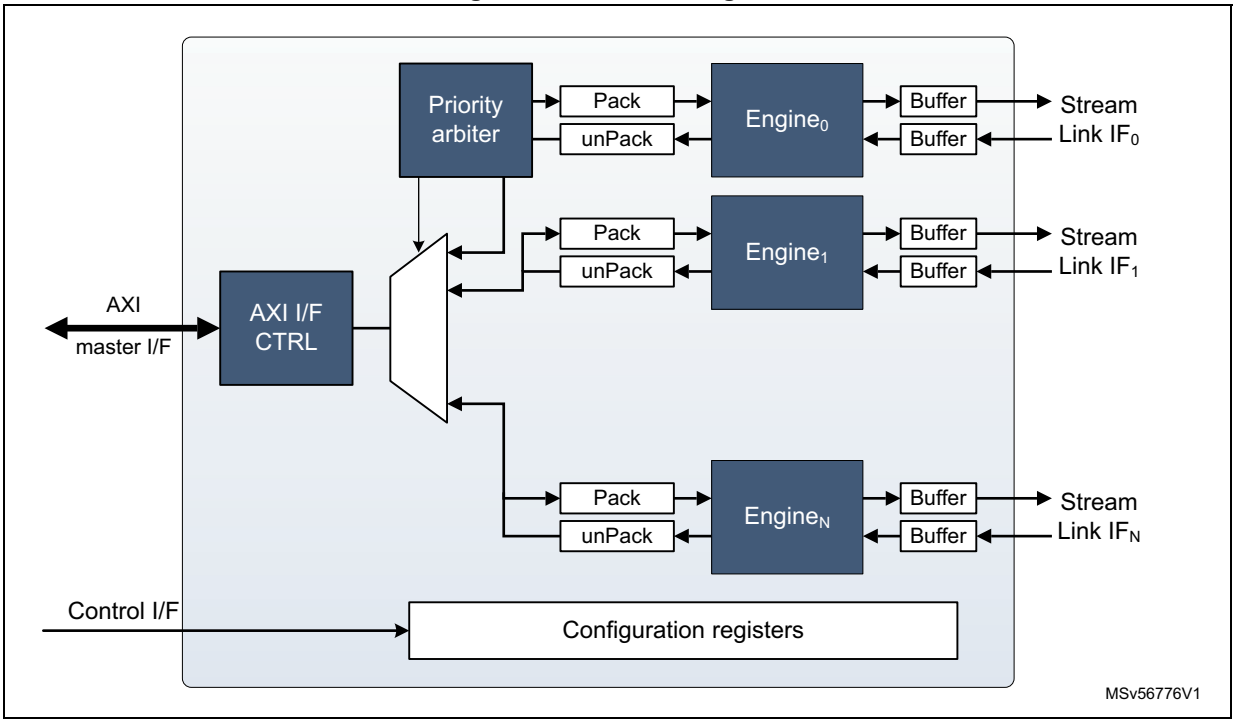
All input stream link ports can forward their data streams to one or multiple (multicast) output ports at the same time, allowing replication of the data stream.

Two stream links can be connected to a single output port of the switch. In this case, the output data are managed using a time-division context switch approach.

14.3.7 Stream engines

These engines manage the transfer from the system to the processing units and vice versa.

Figure 126. Stream engines



A stream engine connects to the stream switch by one input and one output stream link.

The stream engine offers a high configurability on read/write memory patterns:

- 2D and 3D data representations
- Gap management (interleaving)
- High flexibility of data format (size, rounding, subsampling or resampling, MSB or LSB alignment with sign extension).

Linked lists are supported.

14.3.8 Encryption/Decryption unit

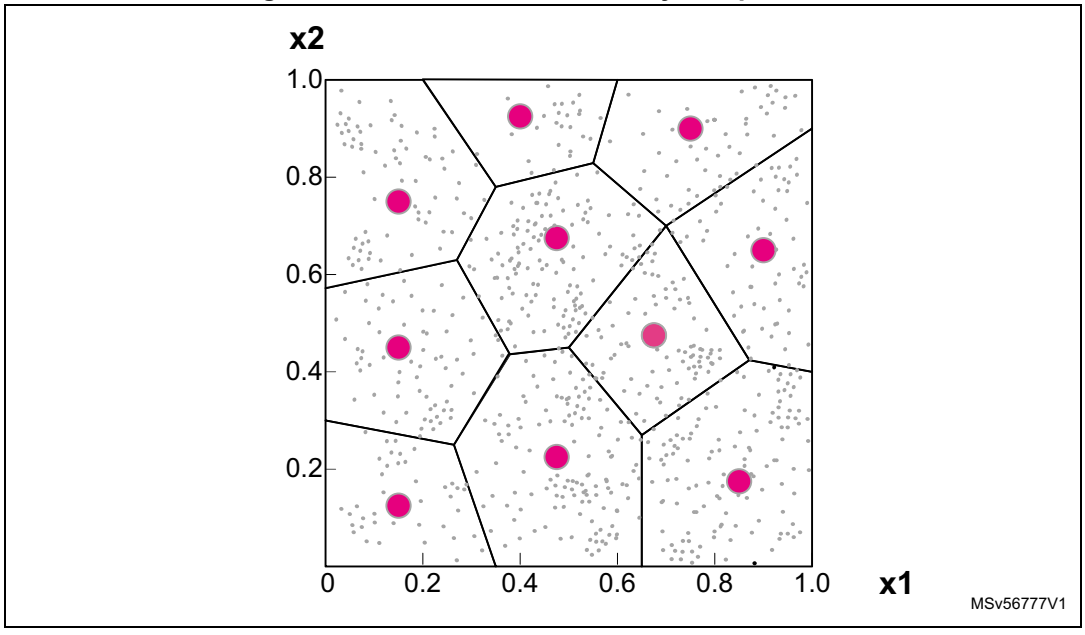
A low overhead/latency encryption/decryption unit based on Keccak-p[200] SHA-3 algorithm cipher, with a programmable number of rounds, is integrated in the bus interface, and can be shared between the different stream engines. It supports both weights and activation decryption and encryption.

14.3.9 Decompression unit

This unit supports on-the-fly decoding for scalar quantized data associated with kernels weights (with arbitrary quantization function), and decoding of vector quantized groups of weights.

An n-dimensional space can be described in a lossy way, by subdividing it into subvolumes, each described by its centroid. The subvolumes are obtained to minimize a predefined distortion function. The whole signal space can be represented with huge savings in terms of storage, as it is possible to represent the space of interest using only centroid values.

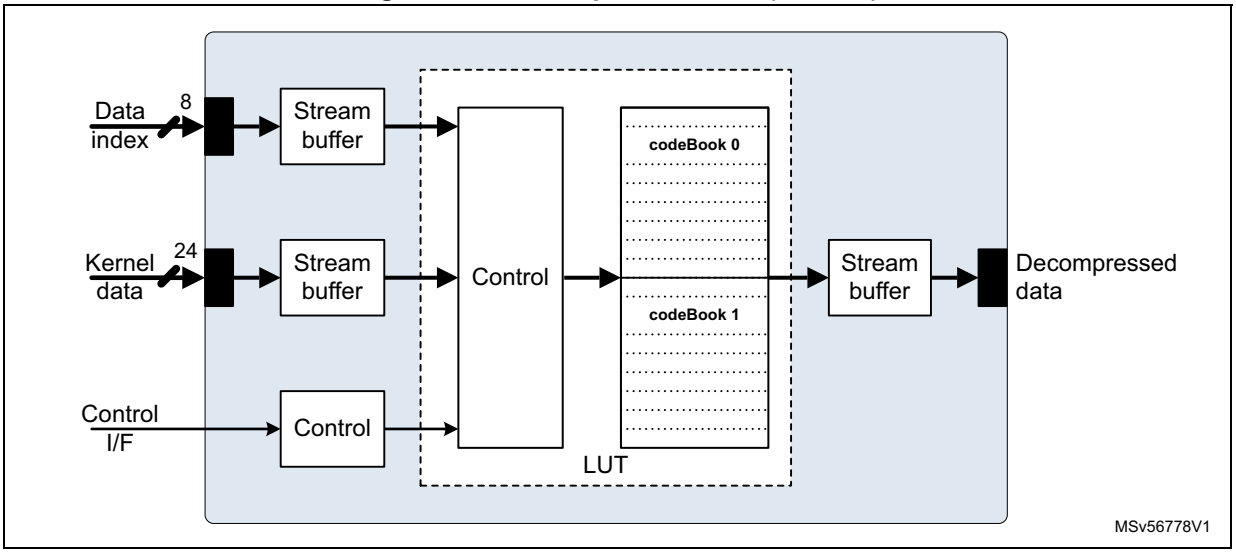
Figure 127. Two dimensional lossy compression



Quantization functions and/or vector quantizer code books (CB) are defined with an offline tool part of the NPU compiler toolchain. The encoder maps the input range into a finite range of rational values, called the code book. Any value stored by the CB is called a code

vector. A code vector is made by up to eight code words. Each code book can store up to 256 code vectors (CV).

Figure 128. Decompression unit (DECUN)



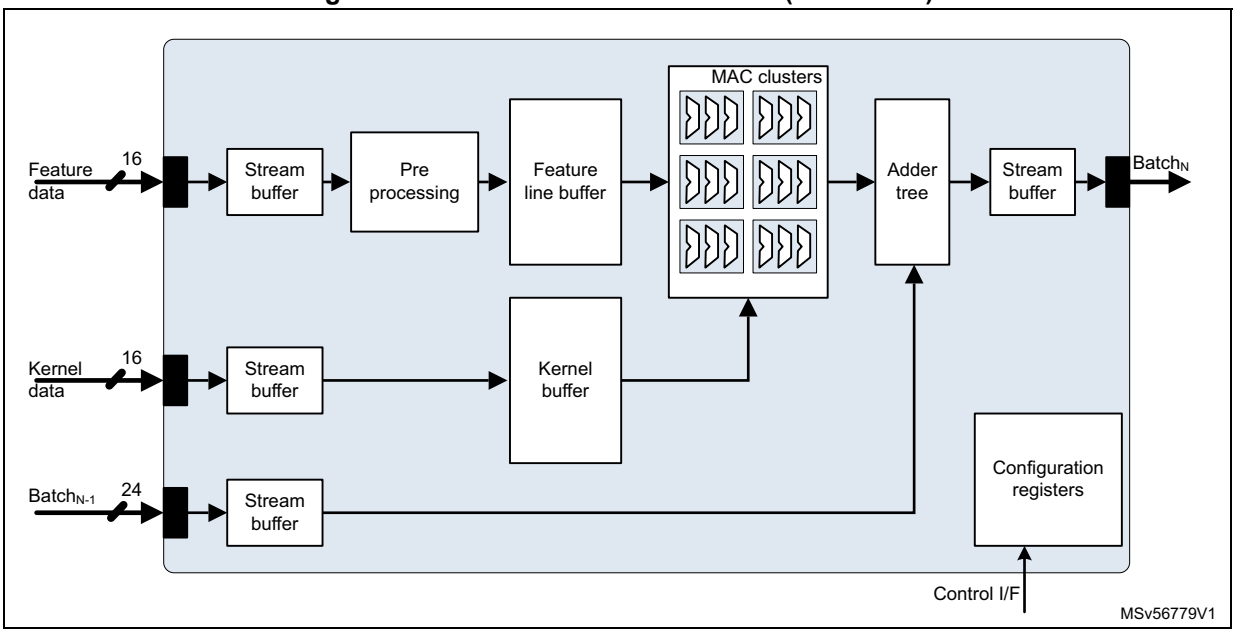
The decompression unit connects to the stream switch by two input stream links and one output stream link.

- iSL_0: receives the 8-bit indexes to decompress the data.
- iSL_1: receives the data to write in the code book, sent as raw data.

14.3.10 Convolutional accelerator

This is the core of the NPU. It implements clusters of multiply-accumulate (MAC) engines supporting fixed-point acceleration of the 3x3 convolutions used by many of the neural network models. It can be configured as 16- or 8-bit MACs, depending on which fixed-point precision for kernels and feature data is considered sufficient.

Figure 129. Convolutional accelerator (CONVACC)



The convolution accelerator connects to the stream switch by three input stream links and one output stream link.

- iSL_0: receives the feature data as 8- or 16-bit raster scan or raw data. Data go through a bit rate adaptation stream buffer and is then transmitted to the feature data preprocessing unit before it is stored in the feature line buffer.
- iSL_1: receives the kernel data as 8- or 16-bit raster scan or raw data.
- oSL: provides convolution results as 8- or 16-bit raster scan or raw data stream, with up to 24-bit wide signed data values.
- iSL_2: can receive intermediate 24-bit signed accumulator values to be accumulated with the convolution results.

Each convolution accelerator supports signed or unsigned arithmetic. It includes eighteen 16-bit MAC units grouped as six clusters of three MAC units. Each MAC performs a single 16x16-, two 16x8-, or four 8x8-bit multiply accumulate operation, which sums up to a maximum of 18 16x16-, 36 16x8-, or 72 8x8-bit multiply accumulate operations per clock cycle and per convolution accelerator unit.

The preprocessing unit implements the following sequence:

1. Format detection: determines if the data is raster scan or raw
2. Shift/round/saturation
3. HOR/VER cropping
4. Zero frame generation: on-the-fly extend the feature tensor structure.
5. Iteration control: simplifies the iterative processing of kernels larger than the 3x3 natively supported by the convolutional accelerator.

14.3.11 Pooling unit

This unit supports pooling operations like local 2D windowed (MxN) min, max, average pooling, as well as global max, min, or average pooling.

Supported features include:

- Native pooling windows of dimensions up to 3x3
- Horizontal and vertical strides ranging from 1 to 15
- Batch size ranging from 1 to 8
- Left, right, and top padding ranging from 0 to 7
- Bottom padding, depending on the window height

Figure 130. Example of pooling operations

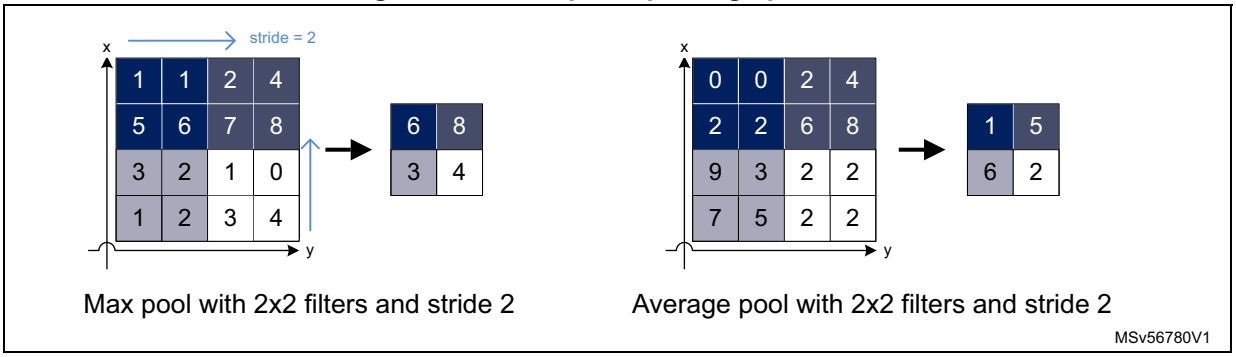
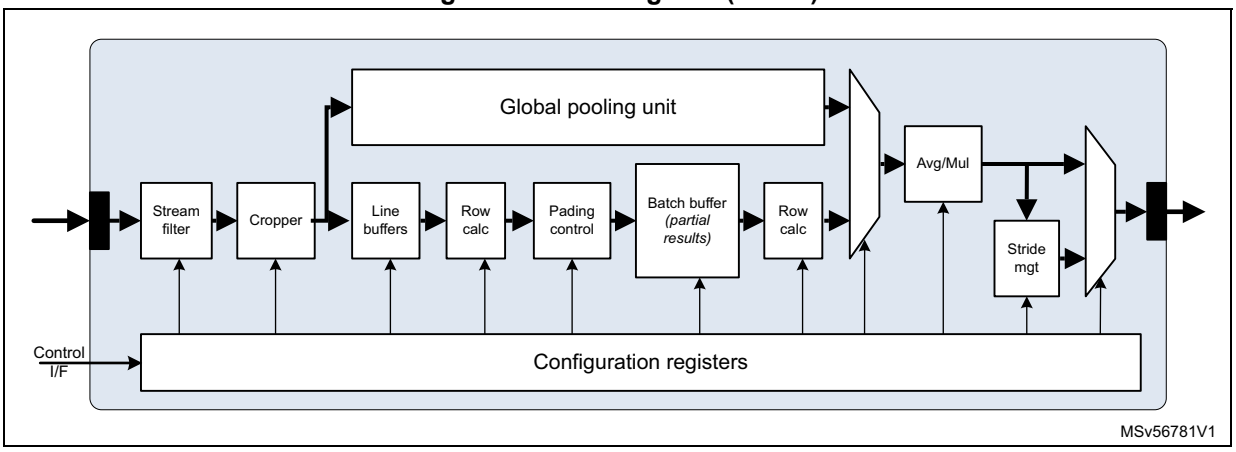


Figure 131. Pooling unit (POOL)



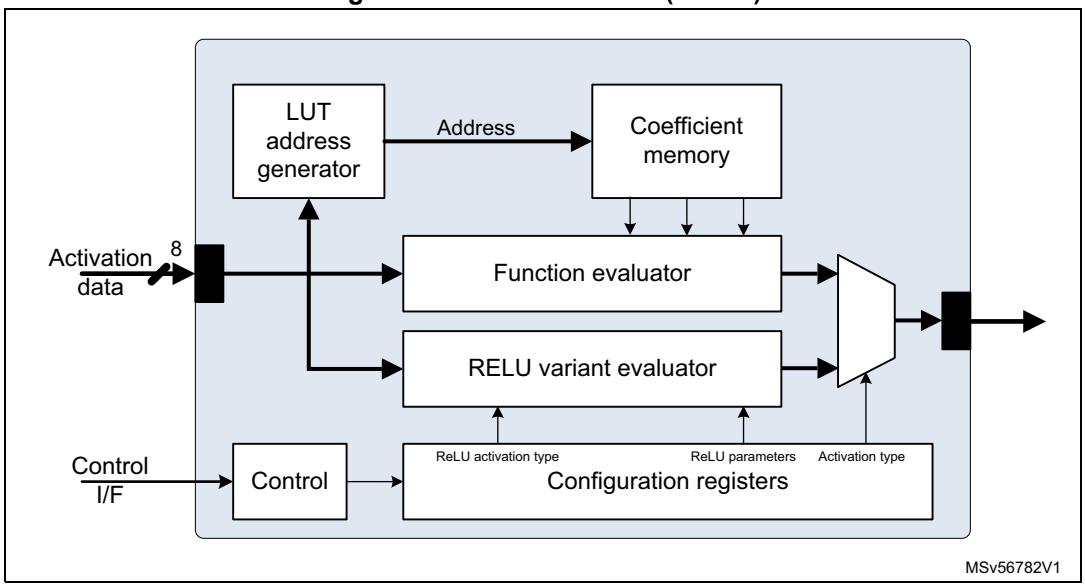
The pooling unit connects to the stream switch by one input and one output stream link:

- iSL_0: receives the activation input values as raster scan or raw data. Input is saturated to 16-bit.
- oSL: produces the pooled output as raw data.

14.3.12 Activation unit

This unit implements the activation functions associated with convolutional neural networks. It provides a dedicated path to compute ReLU-like activations such as plain, parametric, and thresholded ReLUs. There is also a generic function evaluator using a second-degree piece-wise polynomial approximation ($y = ax^2 + bx + c$ or $y = (ax + b) * x + c$). It uses a hierarchical uniform segmentation scheme to implement the evaluator.

Figure 132. Activation unit (ACTIV)



The unit connects to the stream switch by one input stream link and one output stream link.

- iSL_0: receives the activation input values as raster scan or raw data. Input is saturated to 16-bit.
- oSL: produces the result output as raw data.

Table 107 provides examples of usual activation functions.

Table 107. Activation functions example

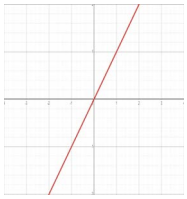
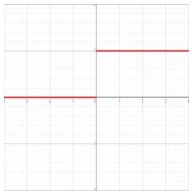
Function	Plot	Equation	Derivative
Identity		$f(x) = x$	$f(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ \infty & \text{for } x = 0 \end{cases}$

Table 107. Activation functions example (continued)

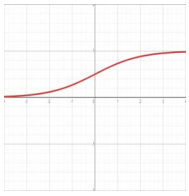
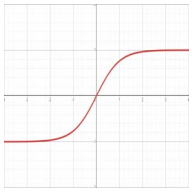
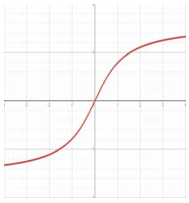
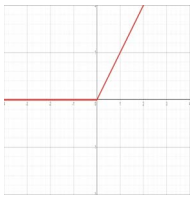
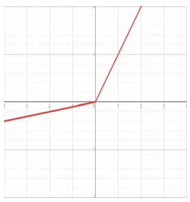
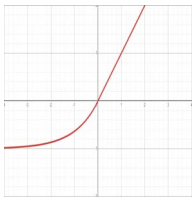
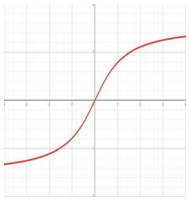
Function	Plot	Equation	Derivative
Logistic		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
Hyperbolic tangent		$f(x) = \tanh(x) = \left(\frac{2}{1+e^{-2x}} - 1\right)$	$f'(x) = 1-f(x)^2$
Arctan		$f(x) = \operatorname{atan}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified linear unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parametric rectified linear unit (PReLU)		$f(x) = \begin{cases} ax & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} a & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential linear unit (ELU)		$f(x) = \begin{cases} a(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + a & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

Table 107. Activation functions example (continued)

Function	Plot	Equation	Derivative
SoftPlus		$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

14.3.13 Arithmetic unit

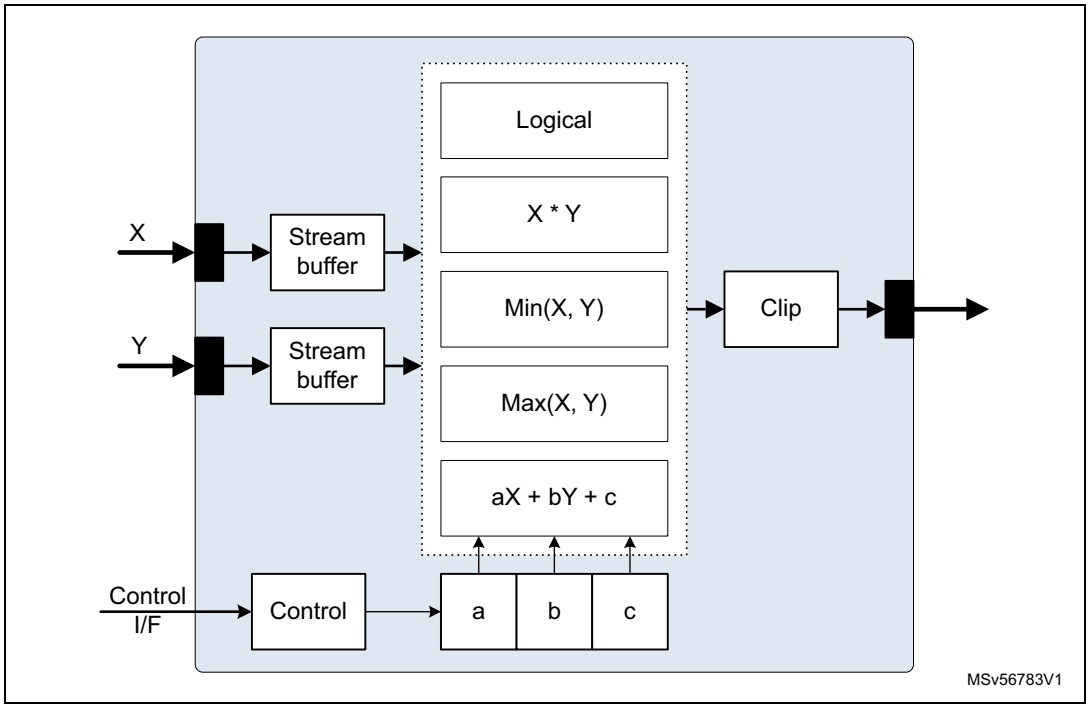
The arithmetic unit can handle outputs coming directly from the convolution accelerators. The main function is aX + bY+ c, where X and Y are input data streams, supplied via two input stream link interfaces and a, b and c are constants that are either vectors or scalars, which are preloaded into an internal memory via the configuration interface.

The arithmetic unit can be configured to perform other operations:

- x * y;
- MIN(x,y), MAX(x,y)
- Logical {AND, OR, NOT(X)}
- Bitwise {AND, OR, XOR, NOT(x)}
- Comparison operations like =, <, ≤, >, ≥
- Math operations like ABS(x), SIGN(x), and CLIP(x) with the 16-bit signed clip ranges supplied by the a coeff (min) and c coeff (max)

Based on the choice and type of coefficients, the unit can also perform element-wise operations like x - y, x + y, and similar ones.

Figure 133. Arithmetic unit (ARITH)



This unit connects to the stream switch by two input stream links and one output stream link.

- $iSL_{0,1}$: receives the input values, denoted X & Y, as raster scan or raw data. Input is saturated to 16-bit.
- oSL: produces the result output as raw data.

14.3.14 Reconfigurable buffer

The amount of data to process is usually divided using channel-wise data segmentation strategies. The data are then organized in execution epochs and channel data blocks along the three spatial dimensions. Data blocks are created considering the number of channels and the size of the incoming data. The partial data coming out from the different data blocks must then be reorganized and used in the next processing.

The reconfigurable buffer takes and stores the input data before making them available to the following units, which is useful to avoid deadlocks.

14.3.15 Epoch controller

This controller makes the NPU operation independent from the host CPU, which just provides a pointer to a 64-bit aligned “binary blob” placed in a memory space accessible through the AXI IF. The binary blob can be encrypted.

The core of the epoch controller is an FSM that decodes the control microinstructions, to appropriately configure all the processing units involved in the model execution using direct access to the NPU internal configuration bus.

The synchronization with the NPU is interrupt-based. An available step-by-step mode allows the execution of one microinstruction at a time. The execution is triggered by a write in a dedicated bit in the IRQ register.

The controller frees the AHB control bus for other uses, and improves the performance through a faster NPU programming. The binary blob is fetched through the AXI interface, which provides a larger bandwidth than the AHB. The direct access to the internal configuration bus allows bypassing all the bus external structure, where transactions from the host processor need to pass through.

14.3.16 Interrupt controller

Functional units can generate level-sensitive, active high, interrupts. These are collected by the interrupt controller, routed to four master interrupt lines, and forwarded to the host CPU.

When a master interrupt fires, the host CPU must read the INTREG register to identify the source of the current interrupt signaling. Interrupt routing to the four master interrupt lines and selective enabling are controlled by the INTORMSK and INTANDMSK mask registers, which have as many bits as interrupt sources.

- INTORMSK: an interrupt is forwarded if any of the unmasked interrupts is active.
- INTANDMSK: an interrupt is forwarded if all the unmasked interrupts are active.

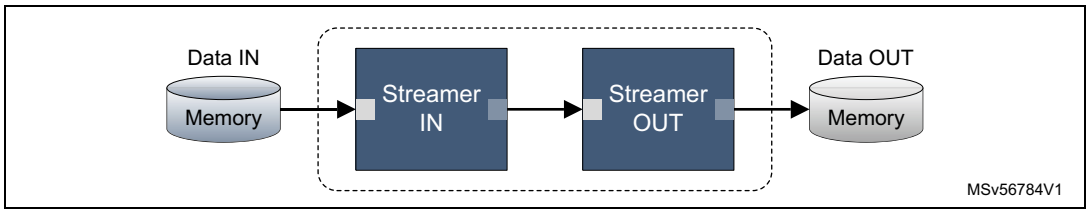
14.4 Configuration examples

The following figures illustrate typical examples of streaming processing supported by the Neural-ART 11 NPU.

14.4.1 DMA transfer

The simplest possible chain involves one input streamer and one output streamer connected back-to-back, which (although not primarily intended for that) configures the NPU as a high-bandwidth DMA.

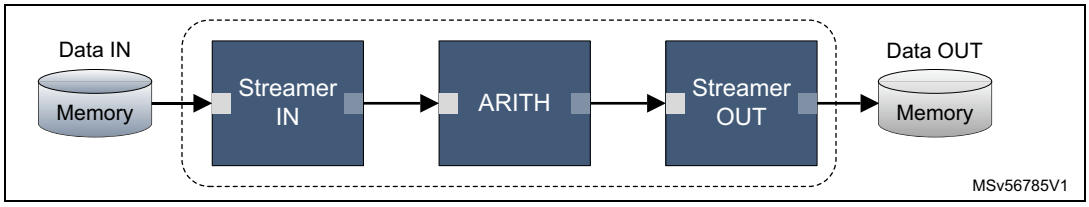
Figure 134. Memory to memory transfer



14.4.2 Simple processing

The simplest processing chain involves one input streamer and one output streamer, with one processing unit in between.

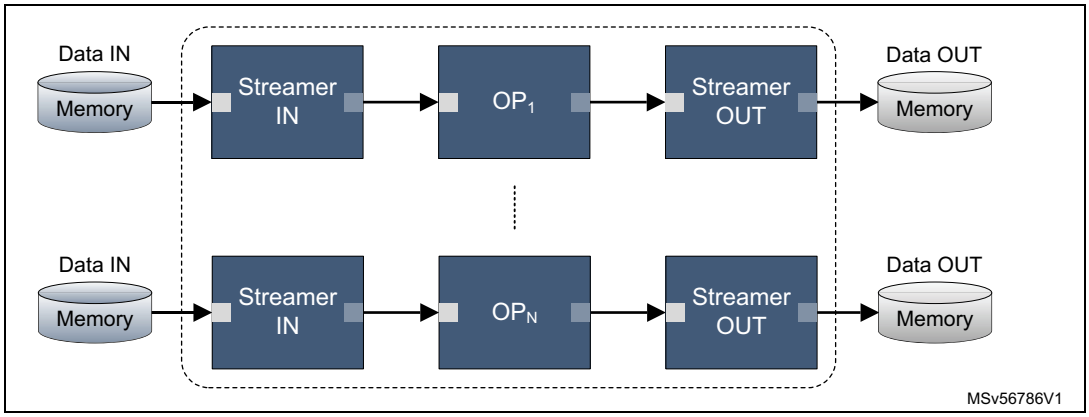
Figure 135. Simple processing



14.4.3 Multiple processing

Implements several independent simple processing chains, like the one in *Figure 135*.

Figure 136. Multiple processing

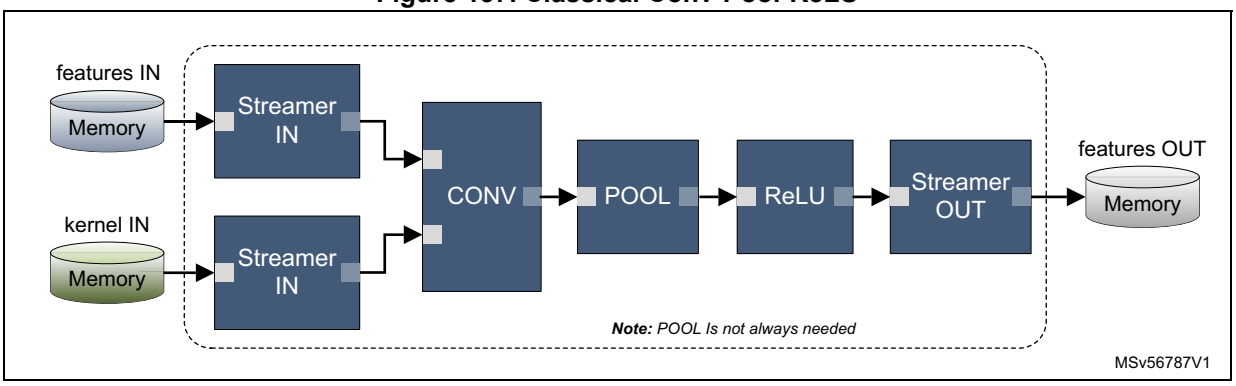


The number of independent chains is limited by the available hardware resources. It cannot exceed half the number of available stream engines, that is, three pairs of stream engines on Neural-ART 11, thus three chains.

14.4.4 Conv-Pool-ReLU

The virtual chain implementing the classical Conv-Pool-ReLU processing is illustrated in *Figure 137*.

Figure 137. Classical Conv-Pool-ReLU



14.5 Address space

The configuration bus address mapping depends on the specific NPU instance configuration. Each of the functional units has a 4-Kbyte configuration space, which yields a 128-Kbyte configuration space for the total IP, organized as shown in *Table 108*.

Table 108. Neural-ART 11 functional units memory base addresses(1)

Offset	Unit	Description
0x0000	CLKCTRL	Clock and reset manager
0x1000	INTCTRL	Interrupt controller
0x2000	BUSIF0	Bus interface 0
0x3000	BUSIF1	Bus interface 1
0x4000	STRSWITCH	Stream switch
0x5000	STRENG0	Stream engine 0
0x6000	STRENG1	Stream engine 1
0x7000	STRENG2	Stream engine 2
0x8000	STRENG3	Stream engine 3
0x9000	STRENG4	Stream engine 4
0xA000	STRENG5	Stream engine 5
0xB000	CONVACC0	Convolutional accelerator unit 0
0xC000	ACTIV0	Activation unit 0
0xD000	ARITH0	Arithmetic unit 0
0xE000	ARITH1	Arithmetic unit 1
0xF000	POOL0	Pooling unit 0
0x10000	EPOCHCTRL0	Epoch controller 0
0x11000	DEBUG0	Debug and trace unit 0

1. The base address is the offset relative to the first address location of the Neural-ART core, defined at integration level. Refer to the memory map.

15 Nested vectored interrupt controller (NVIC)

15.1 NVIC features

The NVIC includes the following features:

- Up to 170 maskable interrupt channels for STM32H7xxx (not including the 10 interrupt lines of Cortex®-M7 with FPU
- 16 programmable priority levels (4 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of system control registers

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

All interrupts, including the core exceptions, are managed by the NVIC.

For more information on exceptions and NVIC programming, refer to PM0253 programming manual for Cortex®-M7.

15.1.1 SysTick

The Cortex®-M7 contains a 24 bit down counter, which can be used as a system timer. The counter is loaded with the RELOAD value programmed in the SYST_RVR register. When enabled, by setting the ENABLE bit in the SYST_CSR register, the counter value decrements every cycle of the SysTick clock. When the counter reaches zero, it is automatically reloaded with the RELOAD value, and generates a SysTick interrupt to the NVIC on IRQ input # -1.

The SysTick clock can be selected by software from either the CPU clock (sys_cpu_ck), or the CPU clock divided by 8, according to the CLK_SOURCE bit in the SYST_CSR register (see *Table 109*).

Table 109. SysTick clock selection

CLK_SOURCE	SysTick clock
0	sys_cpu_ck/8
1	sys_cpu_ck

The value of the SysTick calibration register (SYST_CALIB) is fixed to 1000 (0x3E8). To generate a 1 ms timebase whatever the CPU clock frequency, the SysTick reload value must be programmed as follows in the SYST_RVR register:

- If the SysTick clock source is the CPU clock (sys_cpu_ck):

$$\text{reload value} = (F_{CPU} \times \text{SYST_CALIB}) - 1$$

- If the SysTick clock source is sys_cpu_ck/8:

$$\text{reload value} = ((F_{CPU} / 8) \times \text{SYST_CALIB}) - 1$$

  Where F_{CPU} refers to the CPU clock frequency expressed in MHz.

For example, to obtain a timebase of 1 ms when the SysTick clock source is the CPU clock running at 400 MHz:

$$\text{reload value} = (400 \times \text{SYST_CALIB}) - 1 = 0\text{x}61\text{A}7\text{F}$$

Note: SYST_CALIB has no effect on the SysTick clock frequency. It is provided for informational purposes only.

15.1.2 Interrupt and exception vectors

The exception vectors connected to the NVIC are the following: Reset, NMI, HardFault, MemManage, BusFault, UsageFault, SVCall, DebugMonitor, PendSV, SysTick.

Table 110. NVIC(1)

NVIC position	Priority	Type of priority	Acronym	Description	Address offset
-	-	-	-	Reserved	0x0000 0000
-	1	fixed	Reset	Reset	0x0000 0004
-14	2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (HSE CSS) is linked to the NMI vector.	0x0000 0008
-13	3	fixed	HardFault	All class of fault	0x0000 000C
-12	4	settable	MemManage	Memory management	0x0000 0010
-11	5	settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
-10	6	settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
-	-	-	-	Reserved	0x0000 001C - 0x0000 0028
-5	11	settable	SVCall	System service call via SWI instruction	0x0000 002C
-4	12	settable	DebugMonitor	DebugMonitor	0x0000 0030
-	-	-	-	Reserved	0x0000 0034
-2	14	settable	PendSV	Pendable request for system service	0x0000 0038
-1	15	settable	SysTick	System tick timer	0x0000 003C
0	16	settable	PWR_PVD_AVD	PVD and AVD through the EXTI line	0x0000 0040
-	-	-	-	Reserved	0x0000 0044
2	18	settable	DTS	DTS global interrupt	0x0000 0048
3	19	settable	IWDG	Independent watchdog early warning interrupt	0x0000 004C
4	20	settable	WWDG	Window watchdog early warning interrupt	0x0000 0050
5	21	settable	RCC	RCC global interrupt	0x0000 0054
-	-	-	-	Reserved	0x0000 0058
-	-	-	-	Reserved	0x0000 005C
8	24	settable	FLASH	Flash memory interface	0x0000 0060

Table 110. NVIC(1) (continued)

NVIC position	Priority	Type of priority	Acronym	Description	Address offset
9	25	settable	RAMECC	RAMECC1 OR RAMECC2 interrupt	0x0000 0064
10	26	settable	FPU	FPU safety flags (fpixc/fpidc/fpofc/fpufc/fpdzc/fpioc)	0x0000 0068
-	-	-	-	Reserved	0x0000 006C
12	28	settable	RCC_LSECSS	LSECSS fail interrupt through the EXTI line	0x0000 0070
13	29	settable	TAMP	Tamper interrupt through the EXTI line	0x0000 0074
14	30	settable	CTI0	nIRQ1 from cross trigger	0x0000 0078
15	31	settable	CTI1	nIRQ2 from cross trigger	0x0000 007C
16	32	settable	EXTI0	EXTI Line 0 interrupt through the EXTI line	0x0000 0080
17	33	settable	EXTI1	EXTI Line 1 interrupt through the EXTI line	0x0000 0084
18	34	settable	EXTI2	EXTI Line 2 interrupt through the EXTI line	0x0000 0088
19	35	settable	EXTI3	EXTI Line 3 interrupt through the EXTI line	0x0000 008C
20	36	settable	EXTI4	EXTI Line 4 interrupt through the EXTI line	0x0000 0090
21	37	settable	EXTI5	EXTI Line 5 interrupt through the EXTI line	0x0000 0094
22	38	settable	EXTI6	EXTI Line 6 interrupt through the EXTI line	0x0000 0098
23	39	settable	EXTI7	EXTI Line 7 interrupt through the EXTI line	0x0000 009C
24	40	settable	EXTI8	EXTI Line 8 interrupt through the EXTI line	0x0000 00A0
25	41	settable	EXTI9	EXTI Line 9 interrupt through the EXTI line	0x0000 00A4
26	42	settable	EXTI10	EXTI Line 10 interrupt through the EXTI line	0x0000 00A8
27	43	settable	EXTI11	EXTI Line 11 interrupt through the EXTI line	0x0000 00AC
28	44	settable	EXTI12	EXTI Line 12 interrupt through the EXTI line	0x0000 00B0
29	45	settable	EXTI13	EXTI Line 13 interrupt through the EXTI line	0x0000 00B4
30	46	settable	EXTI14	EXTI Line 14 interrupt through the EXTI line	0x0000 00B8
31	47	settable	EXTI15	EXTI Line 15 interrupt through the EXTI line	0x0000 00BC
32	48	settable	RTC	RTC wake-up OR alarm/SSRU OR time-stamp interrupts through the EXTI lines	0x0000 00C0
-	-	-	-	Reserved	0x0000 00C4
34	50	settable	ADC2	ADC2 global interrupt	0x0000 00C8
35	51	settable	ADC4	ADC4 global interrupt	0x0000 00CC

Table 110. NVIC(1) (continued)

NVIC position	Priority	Type of priority	Acronym	Description	Address offset
36	52	settable	HASH	HASH global interrupt	0x0000 00D0
37	53	settable	RNG	RNG global interrupt	0x0000 00D4
38	54	settable	ADC1	ADC1 global interrupt	0x0000 00D8
39	55	settable	GPDMA1_CH0	GPDMA1 channel 0 interrupt	0x0000 00DC
40	56	settable	GPDMA1_CH1	GPDMA1 channel 1 interrupt	0x0000 00E0
41	57	settable	GPDMA1_CH2	GPDMA1 channel 2 interrupt	0x0000 00E4
42	58	settable	GPDMA1_CH3	GPDMA1 channel 3 interrupt	0x0000 00E8
43	59	settable	GPDMA1_CH4	GPDMA1 channel 4 interrupt	0x0000 00EC
44	60	settable	GPDMA1_CH5	GPDMA1 channel 5 interrupt	0x0000 00F0
45	61	settable	GPDMA1_CH6	GPDMA1 channel 6 interrupt	0x0000 00F4
46	62	settable	GPDMA1_CH7	GPDMA1 channel 7 interrupt	0x0000 00F8
47	63	settable	TIM1_BRK_TERR_IERR	TIM1 break/transition error/index error interrupts	0x0000 00FC
48	64	settable	TIM1_UPD	TIM1 update interrupt	0x0000 0100
49	65	settable	TIM1_TRGI_COM_DIR_IDX	TIM1 trigger/commutation/direction change/index interrupts	0x0000 0104
50	66	settable	TIM1_CC	TIM1 capture compare interrupt	0x0000 0108
51	67	settable	TIM2	TIM2 global interrupt	0x0000 010C
52	68	settable	TIM3	TIM3 global interrupt	0x0000 0110
53	69	settable	TIM4	TIM4 global interrupt	0x0000 0114
54	70	settable	TIM5	TIM5 global interrupt	0x0000 0118
55	71	settable	TIM6	TIM6 global interrupt	0x0000 011C
56	72	settable	TIM7	TIM7 global interrupt	0x0000 0120
57	73	settable	TIM9	TIM9 global interrupt	0x0000 0124
58	74	settable	SPI1	SPI1 global interrupt	0x0000 0128
59	75	settable	SPI2	SPI2 global interrupt	0x0000 012C
60	76	settable	SPI3	SPI3 global interrupt	0x0000 0130
61	77	settable	SPI4	SPI4 global interrupt	0x0000 0134
62	78	settable	SPI5	SPI5 global interrupt	0x0000 0138

Table 110. NVIC(1) (continued)

NVIC position	Priority	Type of priority	Acronym	Description	Address offset
63	79	settable	SPI6	SPI6 global interrupt	0x0000 013C
64	80	settable	HPDMA1_CH0	HPDMA1 channel 0 interrupt	0x0000 0140
65	81	settable	HPDMA1_CH1	HPDMA1 channel 1 interrupt	0x0000 0144
66	82	settable	HPDMA1_CH2	HPDMA1 channel 2 interrupt	0x0000 0148
67	83	settable	HPDMA1_CH3	HPDMA1 channel 3 interrupt	0x0000 014C
68	84	settable	HPDMA1_CH4	HPDMA1 channel 4 interrupt	0x0000 0150
69	85	settable	HPDMA1_CH5	HPDMA1 channel 5 interrupt	0x0000 0154
70	86	settable	HPDMA1_CH6	HPDMA1 channel 6 interrupt	0x0000 0158
71	87	settable	HPDMA1_CH7	HPDMA1 channel 7 interrupt	0x0000 015C
72	88	settable	TIM8_BRK_TERR_IERR	TIM8 break/transition error/index error interrupts	0x0000 0160
73	89	settable	TIM8_UPD	TIM8 update interrupt B	0x0000 0164
74	90	settable	TIM8_TRGI_COM_DIR_IDX	TIM8 trigger/commutation/direction change/index interrupts	0x0000 0168
75	91	settable	TIM8_CC	TIM8 capture compare interrupt	0x0000 016C
76	92	settable	I2C1_EV	I2C1 event interrupt	0x0000 0170
77	93	settable	I2C1_ERR	I2C1 error interrupt	0x0000 0174
78	94	settable	I2C2_EV	I2C2 event interrupt	0x0000 0178
79	95	settable	I2C2_ERR	I2C2 error interrupt	0x0000 017C
80	96	settable	I2C3_EV	I2C3 event interrupt	0x0000 0180
81	97	settable	I2C3_ERR	I2C3 error interrupt	0x0000 0184
82	98	settable	USART1	USART1 global interrupt	0x0000 0188
83	99	settable	USART2	USART2 global interrupt	0x0000 018C
84	100	settable	USART3	USART3 global interrupt	0x0000 0190
85	101	settable	UART4	UART4 global interrupt	0x0000 0194
86	102	settable	UART5	UART5 global interrupt	0x0000 0198
87	103	settable	UART7	UART7 global interrupt	0x0000 019C
88	104	settable	UART8	UART8 global interrupt	0x0000 01A0
89	105	settable	I3C1_EV	I3C1 event interrupt	0x0000 01A4

Table 110. NVIC(1) (continued)

NVIC position	Priority	Type of priority	Acronym	Description	Address offset
90	106	settable	I3C1_ERR	I3C1 error interrupt	0x0000 01A8
-	-	-	-	Reserved	0x0000 01AC
92	108	settable	FMAC1	FMAC1 global interrupt	0x0000 01B0
93	109	settable	CORDIC	CORDIC interrupt	0x0000 01B4
94	110	settable	ADC3	ADC3 global interrupt	0x0000 01B8
95	111	settable	TIM20_BRK_TERR_IERR	TIM20 break/transition error/index error interrupt	0x0000 01BC
96	112	settable	TIM20_UPD	TIM20 update interrupt	0x0000 01C0
97	113	settable	TIM20_TRGI_COM_DIR_IDX	TIM20 trigger/commutation/direction change/index interrupt	0x0000 01C4
98	114	settable	TIM20_CC	TIM20 capture compare interrupt	0x0000 01C8
99	115	settable	TIM10	TIM10 global interrupt	0x0000 01CC
100	116	settable	TIM11	TIM11 global interrupt	0x0000 01D0
101	117	settable	I3C1_WKUP	I3C wake-up Interrupt through EXTI line	0x0000 01D4
102	118	settable	NPU_IT0	NPU end of epoch interrupt	0x0000 01D8
103	119	settable	NPU_IT1	NPU generic interrupt	0x0000 01DC
104	120	settable	NPU_IT2	NPU generic interrupt	0x0000 01E0
105	121	settable	XSPI1	XSPI1 global interrupt	0x0000 01E4
106	122	settable	HSP1	HSP global interrupt	0x0000 01E8
107	123	settable	FMC	FMC global interrupt	0x0000 01EC
108	124	settable	SDMMC1	SDMMC1 global interrupt	0x0000 01F0
109	125	settable	HSP1_ERR	HSP error interrupt	0x0000 01F4
110	126	settable	HSP1_EVT	HSP event interrupt	0x0000 01F8
111	127	settable	HSP1_FWEVT	HSP firmware event interrupt	0x0000 01FC
112	128	settable	OTG_FS	USB OTG FS global interrupt and VBUS interrupt through EXTI lines	0x0000 0200
113	129	settable	TIM12	TIM12 global interrupt	0x0000 0204
114	130	settable	TIM13	TIM13 global interrupt	0x0000 0208
115	131	settable	TIM14	TIM14 global interrupt	0x0000 020C
116	132	settable	TIM15	TIM15 global interrupt	0x0000 0210

Table 110. NVIC(1) (continued)

NVIC position	Priority	Type of priority	Acronym	Description	Address offset
117	133	settable	TIM16	TIM16 global interrupt	0x0000 0214
118	134	settable	TIM17	TIM17 global interrupt	0x0000 0218
119	135	settable	LPTIM1	LPTIM1 global interrupt	0x0000 021C
120	136	settable	LPTIM2	LPTIM2 global interrupt	0x0000 0220
121	137	settable	LPTIM3	LPTIM3 global interrupt	0x0000 0224
122	138	settable	LPTIM4	LPTIM4 global interrupt	0x0000 0228
123	139	settable	LPTIM5	LPTIM5 global interrupt	0x0000 022C
124	140	settable	UART9	UART9 global interrupt	0x0000 0230
125	141	settable	USART10	USART10 global interrupt	0x0000 0234
126	142	settable	ADF1_FLT0	ADF1 filter 0 global interrupt	0x0000 0238
127	143	settable	CRS	CRS global interrupt	0x0000 023C
128	144	settable	UCPD1	UCPD global interrupt	0x0000 0240
129	145	settable	WKUP5678	Interrupt for 4 wake-up pins (5, 6, 7, 8) through EXTI line	0x0000 0244
130	146	settable	USART6	USART6 global interrupt	0x0000 0248
131	147	settable	LPUART1	LPUART1 global interrupt	0x0000 024C
132	148	settable	WKUP1234	Interrupt for 4 wake-up pins (1, 2, 3, 4) through EXTI line	0x0000 0250
133	149	settable	GPDMA1_CH8	GPDMA1 channel 8 interrupt	0x0000 0254
134	150	settable	GPDMA1_CH9	GPDMA1 channel 9 interrupt	0x0000 0258
135	151	settable	GPDMA1_CH10	GPDMA1 channel 10 interrupt	0x0000 025C
136	152	settable	GPDMA1_CH11	GPDMA1 channel 11 interrupt	0x0000 0260
137	153	settable	GPDMA1_CH12	GPDMA1 channel 12 interrupt	0x0000 0264
138	154	settable	GPDMA1_CH13	GPDMA1 channel 13 interrupt	0x0000 0268
139	155	settable	GPDMA1_CH14	GPDMA1 channel 14 interrupt	0x0000 026C
140	156	settable	GPDMA1_CH15	GPDMA1 channel 15 interrupt	0x0000 0270
141	157	settable	HPDMA1_CH8	HPDMA1 channel 8 interrupt	0x0000 0274
142	158	settable	HPDMA1_CH9	HPDMA1 channel 9 interrupt	0x0000 0278
143	159	settable	HPDMA1_CH10	HPDMA1 channel 10 interrupt	0x0000 027C

Table 110. NVIC(1) (continued)

NVIC position	Priority	Type of priority	Acronym	Description	Address offset
144	160	settable	HPDMA1_CH11	HPDMA1 channel 11 interrupt	0x0000 0280
145	161	settable	HPDMA1_CH12	HPDMA1 channel 12 interrupt	0x0000 0284
146	162	settable	HPDMA1_CH13	HPDMA1 channel 13 interrupt	0x0000 0288
147	163	settable	HPDMA1_CH14	HPDMA1 channel 14 interrupt	0x0000 028C
148	164	settable	HPDMA1_CH15	HPDMA1 channel 15 interrupt	0x0000 0290
-	-	-	-	Reserved	0x0000 0294
150	166	settable	FDCAN3_IT0	FDCAN3 interrupt 0	0x0000 0298
151	167	settable	FDCAN3_IT1	FDCAN3 interrupt 1	0x0000 029C
152	168	settable	FDCAN1_IT0	FDCAN1 interrupt 0	0x0000 02A0
153	169	settable	FDCAN1_IT1	FDCAN1 interrupt 1	0x0000 02A4
154	170	settable	FDCAN2_IT0	FDCAN2 interrupt 0	0x0000 02A8
155	171	settable	FDCAN2_IT1	FDCAN2 interrupt 1	0x0000 02AC
156	172	settable	DAC1	DAC1 underrun interrupt	0x0000 02B0
157	173	settable	DAC2	DAC2 underrun interrupt	0x0000 02B4
158	174	settable	DAC3	DAC3 underrun interrupt	0x0000 02B8
159	175	settable	DAC4	DAC4 underrun interrupt	0x0000 02BC
160	176	settable	COMP1	COMP1 output interrupt through EXTI line	0x0000 02C0
161	177	settable	COMP2	COMP2 output interrupt through EXTI line	0x0000 02C4
162	178	settable	COMP3	COMP3 output interrupt through EXTI line	0x0000 02C8
163	179	settable	COMP4	COMP4 output interrupt through EXTI line	0x0000 02CC
164	180	settable	HRTIM1_M	HRtimer1 master interrupt	0x0000 02D0
165	181	settable	HRTIM1_TIMA	HRtimer1 slave A interrupt	0x0000 02D4
166	182	settable	HRTIM1_TIMB	HRtimer1 slave B interrupt	0x0000 02D8
167	183	settable	HRTIM1_TIMC	HRtimer1 slave C interrupt	0x0000 02DC
168	184	settable	HRTIM1_TIMD	HRtimer1 slave D interrupt	0x0000 02E0
169	185	settable	HRTIM1_TIME	HRtimer1 slave E interrupt	0x0000 02E4
170	186	settable	HRTIM1_TIMF	HRtimer1 slave F interrupt	0x0000 02E8
171	187	settable	HRTIM1_FLT	HRtimer1 error interrupt	0x0000 02EC
172	188	settable	HRTIM2_M	HRtimer2 master interrupt	0x0000 02F0

Table 110. NVIC(1) (continued)

NVIC position	Priority	Type of priority	Acronym	Description	Address offset
173	189	settable	HRTIM2_TIMA	HRtimer2 slave A interrupt	0x0000 02F4
174	190	settable	HRTIM2_TIMB	HRtimer2 slave B interrupt	0x0000 02F8
175	191	settable	HRTIM2_TIMC	HRtimer2 slave C interrupt	0x0000 02FC
176	192	settable	HRTIM2_TIMD	HRtimer2 slave D interrupt	0x0000 0300
177	193	settable	HRTIM2_TIME	HRtimer2 slave E interrupt	0x0000 0304
178	194	settable	HRTIM2_TIMF	HRtimer2 slave F interrupt	0x0000 0308
179	195	settable	HRTIM2_FLT	HRtimer2 error interrupt	0x0000 030C
180	196	settable	COMP5	COMP5 output interrupt through EXTI line	0x0000 0310
181	197	settable	COMP6	COMP6 output interrupt through EXTI line	0x0000 0314
182	198	settable	COMP7	COMP7 output interrupt through EXTI line	0x0000 0318
183	199	settable	COMP8	COMP8 output interrupt through EXTI line	0x0000 031C

1. When different signals are connected to the same NVIC interrupt line, they are OR-ed.

16 Extended interrupt and event controller (EXTI)

The extended interrupt and event controller (EXTI) manages wake-up through configurable and direct event inputs. It provides wake-up requests to the power control, and generates interrupt requests to the CPU NVIC and events to the CPU event input.

The EXTI wake-up requests allow the system to be woken up from stop mode, and the CPU to be woken up from CStop mode.

Both the interrupt request and event request generation can also be used in run modes.

16.1 EXTI main features

The EXTI main features are as follows:

- All event inputs allow the CPU to wake up and to generate a CPU interrupt and/or CPU event

The asynchronous event inputs are classified in two groups:

- Configurable events (signals from I/Os or peripherals able to generate a pulse) that feature:
  - Selectable active trigger edge
  - Interrupt pending status register bit
  - Individual interrupt and event generation mask
  - Software trigger possibility
- Direct events (interrupt and wake-up sources from other peripherals, requiring to be cleared in the peripheral) that feature:
  - Fixed rising edge active trigger
  - No interrupt pending status register bit in the EXTI (the interrupt pending status is provided by the peripheral generating the event)
  - Individual interrupt and event generation mask
  - No software trigger possibility.

16.2 EXTI block diagram

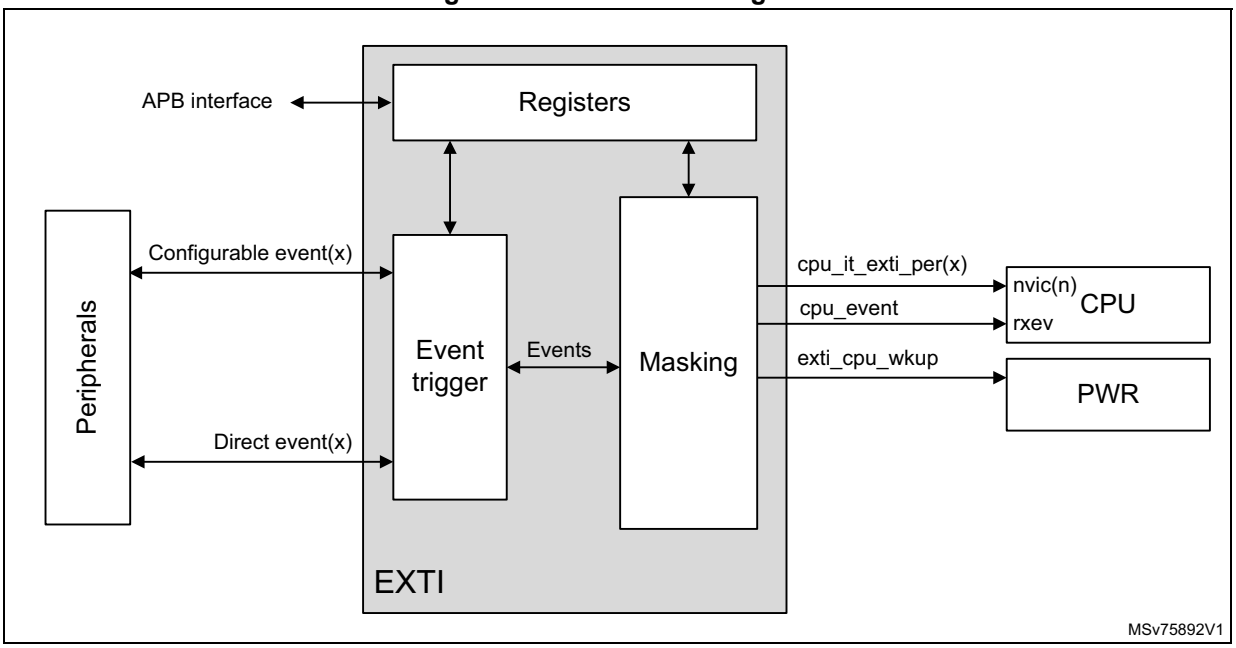
As shown in *Figure 138*, the EXTI consists of a register block accessed via an APB interface, an event input trigger block, and a masking block.

The register block contains all EXTI registers.

The event input trigger block provides event input edge triggering logic.

The masking block provides the event input distribution to the different wake-up, interrupt and event outputs, and their masking.

Figure 138. EXTI block diagram



16.2.1 EXTI connections between peripherals and CPU

The peripherals able to generate wake-up events when the system is in Stop mode or the CPU is in CStop mode are connected to an EXTI configurable event input or direct event input:

- Peripheral signals that generate a pulse are connected to an EXTI configurable event input. For these events, the EXTI provides a CPU status pending bit that has to be cleared.
- Peripheral interrupt and wake-up sources that have to be cleared in the peripheral are connected to an EXTI direct event input. There is no CPU status pending bit within the EXTI. The interrupt or wake-up is cleared by the CPU in the peripheral.

The CPU interrupts are connected to their respective CPU NVIC, and, similarly, the CPU event is connected to the CPU rxev input.

The EXTI wake-up signals are connected to the PWR block, and are used to wake up the CPU.

16.3 EXTI functional description

Depending on the EXTI event input type and wake-up target(s), different logic implementations are used. The applicable features are controlled from register bits:

- Active trigger edge enable, by *EXTI rising trigger selection register 1 (EXTI_RTSR1)*, *EXTI rising trigger selection register 2 (EXTI_RTSR2)*, *EXTI rising trigger selection register 3 (EXTI_RTSR3)*, and *EXTI falling trigger selection register 1 (EXTI_FTSR1)*, *EXTI falling trigger selection register 2 (EXTI_FTSR2)*, *EXTI falling trigger selection register 3 (EXTI_FTSR3)*
- Software trigger, by *EXTI software interrupt event register 1 (EXTI_SWIER1)*, *EXTI software interrupt event register 2 (EXTI_SWIER2)*, *EXTI software interrupt event register 3 (EXTI_SWIER3)*
- CPU interrupt enable, by *EXTI interrupt mask register 1 (EXTI_IMR1)*, *EXTI interrupt mask register 2 (EXTI_IMR2)*, *EXTI interrupt mask register 3 (EXTI_IMR3)*
- CPU event enable, by *EXTI event mask register 1 (EXTI_EMR1)*, *EXTI event mask register 2 (EXTI_EMR2)*, *EXTI event mask register 3 (EXTI_EMR3)*

Table 111. EXTI event input configurations and register control(1)

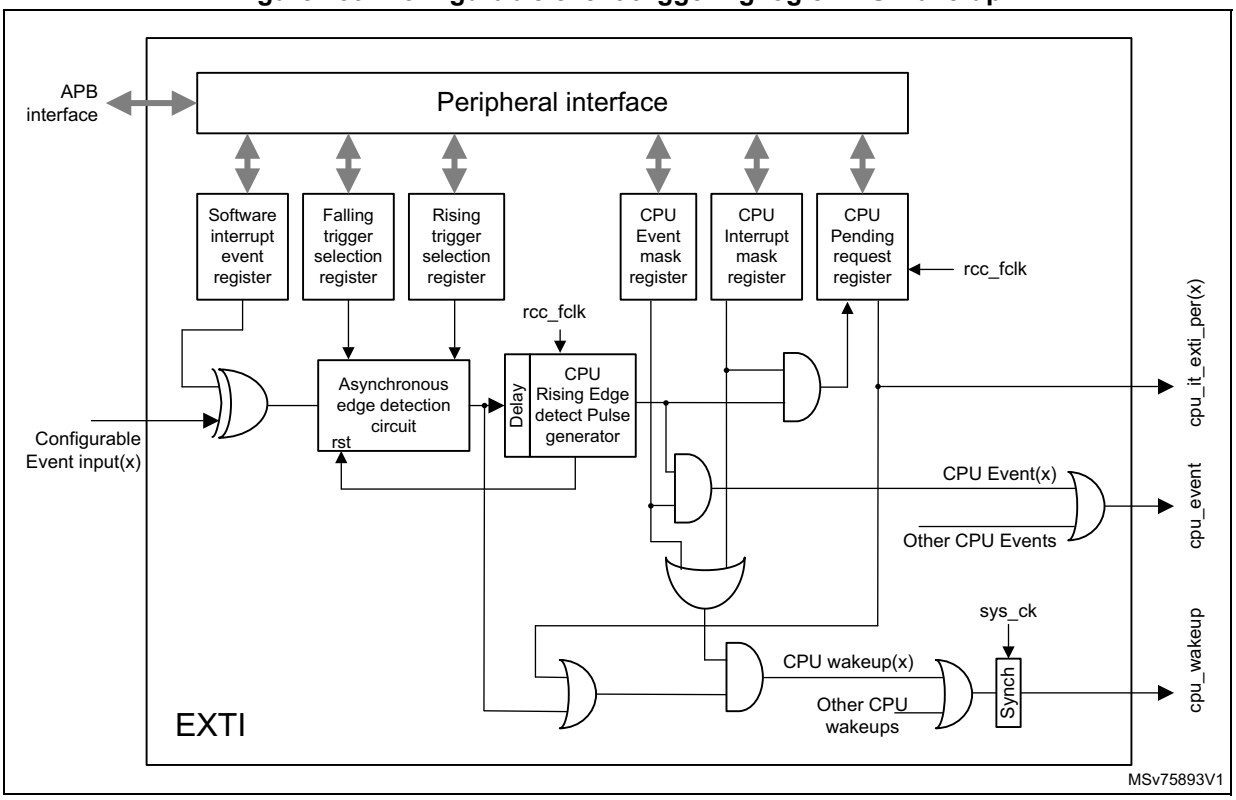
Event input type	Wake-up target(s)	Logic implementation	EXTI_RTSR	EXTI_FTSR	EXTI_SWIER	EXTI_IMR	EXTI_EMR
Configurable	CPU	Configurable event input, CPU wake-up logic	X	X	X	X	X
Direct	CPU	Direct event input, CPU wake-up logic	-	-	-	X	X

1. X indicates that functionality is available.

16.3.1 EXTI configurable event input - CPU wake-up

Figure 139 is a detailed representation of the logic associated with configurable event inputs which always wake up the CPU.

Figure 139. Configurable event triggering logic CPU wake-up



The software interrupt event register allows the system to trigger configurable events by software by writing the *EXTI software interrupt event register 1 (EXTI_SWIER1)*, *EXTI software interrupt event register 2 (EXTI_SWIER2)* and *EXTI software interrupt event register 3 (EXTI_SWIER3)*.

The rising edge *EXTI rising trigger selection register 1 (EXTI_RTSR1)*, *EXTI rising trigger selection register 2 (EXTI_RTSR2)*, *EXTI rising trigger selection register 3 (EXTI_RTSR3)*, and falling edge *EXTI falling trigger selection register 1 (EXTI_FTSR1)*, *EXTI falling trigger selection register 2 (EXTI_FTSR2)*, *EXTI falling trigger selection register 3 (EXTI_FTSR3)* selection registers allow the system to enable and select the configurable event active trigger edge or both edges.

The device features dedicated interrupt mask registers, namely *EXTI interrupt mask register 1 (EXTI_IMR1)*, *EXTI interrupt mask register 2 (EXTI_IMR2)*, and *EXTI interrupt mask register 3 (EXTI_IMR3)*, together with *EXTI pending register 1 (EXTI_PR1)*, *EXTI pending register 2 (EXTI_PR2)* and *EXTI pending register 3 (EXTI_PR3)* configurable event request pending registers. The pending register bit is only set for an unmasked interrupt. Each event generates an individual interrupt to the CPU NVIC. The configurable event interrupts need to be acknowledged (reset) by software in the EXTI_PRn register.

The device features dedicated event mask registers, that is, *EXTI event mask register 1 (EXTI_EMR1)*, *EXTI event mask register 2 (EXTI_EMR2)*, and *EXTI event mask register 3 (EXTI_EMR3)*. The enabled event then generates an event on the CPU. All events for a CPU are OR-ed together into a single CPU event signal. The CPU pending register (EXTI_PRn) is not set for an unmasked CPU event.

When a CPU interrupt or CPU event is enabled, the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the CPU clock is woken up before the asynchronous edge detection circuit is reset.

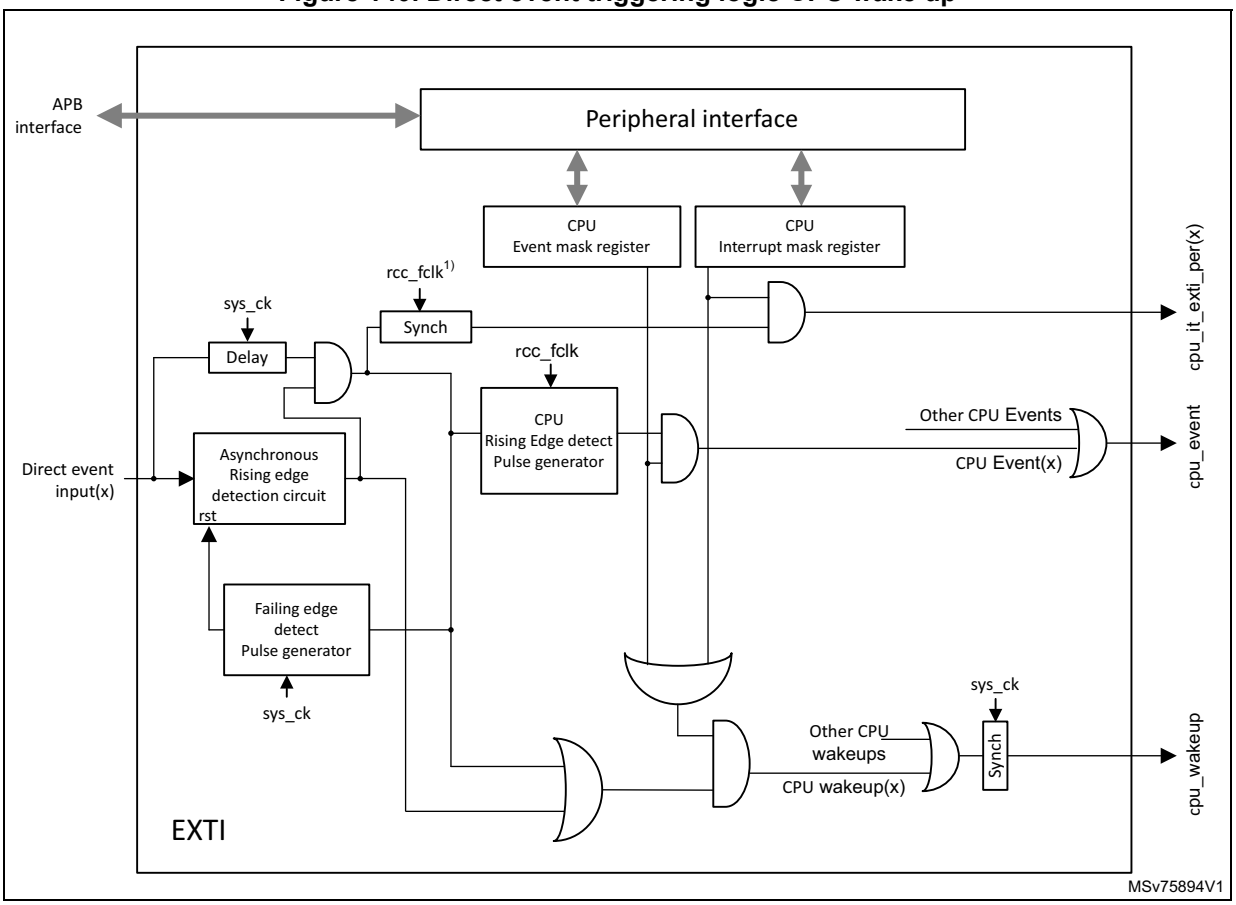
Note: A detected configurable event, enabled by the CPU, is only cleared when the CPU wakes up.

16.3.2 EXTI direct event input - CPU wake up

Figure 140 is a detailed representation of the logic associated with direct event inputs waking up the CPU.

Direct events only provide CPU interrupt enable and CPU event enable functionality.

Figure 140. Direct event triggering logic CPU wake up



1. The CPU interrupt for asynchronous direct event inputs (peripheral wake-up signals) is synchronized with the CPU clock. The synchronous direct event inputs (peripheral interrupt signals), after the asynchronous edge detection, are directly sent to the CPU interrupt without resynchronization.

16.4 EXTI event input mapping

For the 16 GPIO event inputs, the associated IOPORT pin has to be selected in the SBS register SBS_EXTICRn. The same pin from each IOPORT maps to the corresponding EXTI event input.

The wake-up capabilities of each event input are detailed in *Table 112*. An event input can wake up the CPU.

The EXTI event inputs with a connection to the CPU NVIC are indicated in the “Connection to NVIC” column. For the EXTI events not having a connection to the NVIC, the peripheral interrupt is directly connected to the NVIC in parallel with the connection to the EXTI.

All EXTI event inputs are OR-ed together and connected to the CPU event input (rxev).

Table 112. EXTI event input mapping

Event input	Source	Event input type	Connection to NVIC
0 - 15	EXTI[15:0]	Configurable	Yes
16	PVD and AVD(1)	Configurable	Yes
17	RTC alarm, SSRU	Configurable	Yes
18	RTC timestamp	Configurable	Yes
19	RTC wake-up timer	Configurable	Yes
20	USB_OTG_FS VBUS plug/unplug	Configurable	Yes
21	TAMP wake-up	Configurable	Yes
22	I2C1 wake-up	Direct	Yes
23	I2C2 wake-up	Direct	Yes
24	I2C3 wake-up	Direct	Yes
25	I3C1 wake-up	Direct	Yes
26	USART1 wake-up	Direct	Yes
27	USART2 wake-up	Direct	Yes
28	USART3 wake-up	Direct	Yes
29	USART6 wake-up	Direct	Yes
30	UART4 wake-up	Direct	Yes
31	UART5 wake-up	Direct	Yes
32	UART7 wake-up	Direct	Yes
33	UART8 wake-up	Direct	Yes
34	ADF wake-up	Configurable	Yes
35	LPUART1 wake-up	Direct	Yes
36	SPI1 wake-up	Direct	Yes
37	SPI2 wake-up	Direct	Yes
38	SPI3 wake-up	Direct	Yes
39	SPI4 wake-up	Direct	Yes

Table 112. EXTI event input mapping (continued)

Event input	Source	Event input type	Connection to NVIC
40	SPI5 wake-up	Direct	Yes
41	SPI6 wake-up	Direct	Yes
42	UART9 wake-up	Direct	Yes
43	USB_OTG_FS wake-up	Direct	Yes
44	USART10 wake-up	Direct	Yes
45	UCPD wake-up	Direct	Yes
46	LSECSS failure wake-up	Configurable	Yes
47	LPTIM1 wake-up	Direct	Yes
48	LPTIM2 wake-up	Direct	Yes
49	LPTIM2 CH1 output	Configurable	No
50	LPTIM3 wake-up	Direct	Yes
51	LPTIM3 CH1 output	Configurable	No
52	LPTIM4 wake-up	Direct	Yes
53	LPTIM5 wake-up	Direct	Yes
54	Reserved	-	-
55	WKUP1	Direct	Yes
56	WKUP2	Direct	Yes
57	WKUP3	Direct	Yes
58	WKUP4	Direct	Yes
59	IWDG wake-up	Direct	Yes
60	COMP1	Configurable	Yes
61	COMP2	Configurable	Yes
62	COMP3	Configurable	Yes
63	COMP4	Configurable	Yes
64	WKUP5	Direct	Yes
65	WKUP6	Direct	Yes
66	WKUP7	Direct	Yes
67	WKUP8	Direct	Yes
68	COMP5	Configurable	Yes
69	COMP6	Configurable	Yes
70	COMP7	Configurable	Yes
71	COMP8	Configurable	Yes
72	Reserved	-	-
73	Reserved	-	-
74	Reserved	-	-

Table 112. EXTI event input mapping (continued)

Event input	Source	Event input type	Connection to NVIC
75	Reserved	-	-
76	Reserved	-	-
77	DTS wake-up	Direct	Yes
78	Reserved	-	-
79	Reserved	-	-
80	Reserved	-	-

1. PVD and AVD signals are OR-ed together on the same EXTI event input.

16.5 EXTI functional behavior

The direct event inputs are enabled in the respective peripheral generating the event. The configurable events are enabled by enabling at least one of the trigger edges.

An event only wakes up the CPU when the event-associated CPU interrupt is unmasked and/or the CPU event is unmasked.

Table 113. Masking functionality

CPU		Configurable event inputs PRx bits of EXTI_PR	CPU		
Interrupt enable MRx bits of EXTI_IMR	Event enable MRx bits of EXTI_EMR		Interrupt	Event	Wake-up
0	0	No	Masked	Masked	Masked
0	1	No	Masked	Yes	Yes
1	0	Status latched	Yes	Masked	Yes
1	1	Status latched	Yes	Yes	Yes

For configurable event inputs, when the enabled edge(s) occur on the event input, an event request is generated. When the associated CPU interrupt is unmasked, the corresponding pending PRx bit in EXTI_PR is set and the CPU interrupt signal is activated. EXTI_PR PRx pending bit must be cleared by software writing it to 1. This clears the CPU interrupt.

For direct event inputs, when enabled in the associated peripheral, an event request is generated on the rising edge only. There is no corresponding CPU pending bit. When the associated CPU interrupt is unmasked the corresponding CPU interrupt signal is activated.

The CPU event has to be unmasked to generate an event. When the enabled edge(s) occur on the event input, a CPU event pulse is generated. There is no CPU event pending bit.

Both a CPU interrupt and a CPU event may be enabled on the same event input. They both trigger the same event input condition(s).

For the configurable event inputs, an event input request can be generated by software when writing a 1 in the software interrupt/event register EXTI_SWIER.

16.5.1 EXTI CPU interrupt procedure

- Unmask the event input interrupt by setting the corresponding mask bits in the EXTI_IMR register.
- For configurable event inputs, enable the event input by setting either one or both corresponding trigger edge enable bits in EXTI_RTSR and EXTI_FTSR registers.
- Enable the associated interrupt source in the CPU NVIC or use the SEVONPEND, so that an interrupt coming from the CPU interrupt signal is detectable by the CPU after a WFI/WFE instruction.
  - For configurable event inputs the associated EXTI pending bit needs to be cleared.

16.5.2 EXTI CPU event procedure

- Unmask the event input by setting the corresponding mask bits of the EXTI_EMR register.
- For configurable event inputs, enable the event input by setting either one or both corresponding trigger edge enable bits in EXTI_RTSR and EXTI_FTSR registers.
- The CPU event signal is detected by the CPU after a WFE instruction.
  - For configurable event inputs there is no EXTI pending bit to clear.

16.5.3 EXTI CPU wake-up procedure

- Unmask the event input by setting at least one of the corresponding mask bits in the EXTI_IMR and/or EXTI_EMR registers. The CPU wake-up is generated at the same time as the unmasked CPU interrupt and/or CPU event.
- For configurable event inputs, enable the event input by setting either one or both corresponding trigger edge enable bits in the EXTI_RTSR and EXTI_FTSR registers.
- Direct events automatically generate a CPU wake-up.

16.5.4 EXTI software interrupt/event trigger procedure

Any of the configurable event inputs can be triggered from the software interrupt/event register (the associated CPU interrupt and/or CPU event must be enabled by their respective procedure).

- Enable the event input by setting at least one of the corresponding edge trigger bits in the EXTI_RTSR and/or EXTI_FTSR registers.
- Unmask the software interrupt/event trigger by setting at least one of the corresponding mask bits in the EXTI_IMR and/or EXTI_EMR registers.
- Trigger the software interrupt/event by writing 1 to the corresponding bit in the EXTI_SWIER register.
- The event input may be disabled by clearing the EXTI_RTSR and EXTI_FTSR register bits.

Note: An edge on the configurable event input also triggers an interrupt/event.

16.6 EXTI registers

The registers can only be accessed with a 32-bit (word). A byte or half-word cannot be read or written.

16.6.1 EXTI rising trigger selection register 1 (EXTI_RTSR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT21	RT20	RT19	RT18	RT17	RT16
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 21:0 **RTi:** Rising trigger event configuration bit of configurable event input i (i = 21 to 0)(1)

0: Rising trigger disabled (for event and interrupt) for input line i

1: Rising trigger enabled (for event and interrupt) for input line i

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs while the register is being written, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

16.6.2 EXTI falling trigger selection register 1 (EXTI_FTSR1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT21	FT20	FT19	FT18	FT17	FT16
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 21:0 **FTi:** Falling trigger event configuration bit of configurable event input i (i = 21 to 0)(1)

0: Falling trigger disabled (for event and interrupt) for input line i

1: Falling trigger enabled (for event and interrupt) for input line i

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs while the register is being written, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

16.6.3 EXTI software interrupt event register 1 (EXTI_SWIER1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW21	SW20	SW19	SW18	SW17	SW16
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW15	SW14	SW13	SW12	SW11	SW10	SW9	SW8	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 21:0 **SWi:** Software interrupt on event i (i = 21 to 0)
This bitfield always returns 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit triggers an event on line i. This bit is auto cleared by hardware.

16.6.4 EXTI rising trigger selection register 2 (EXTI_RTSR2)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RT63	RT62	RT61	RT60	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT51	Res.	RT49	Res.
rw	rw	rw	rw									rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RT46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT34	Res.	Res.
	rw												rw		

Bits 31:28 **RTi:** Rising trigger event configuration bit of configurable event input i (i = 63 to 60)(1)
0: Rising trigger disabled (for event and interrupt) for input line i
1: Rising trigger enabled (for event and interrupt) for input line i

Bits 27:20 Reserved, must be kept at reset value.

Bit 19 **RT51:** Rising trigger event configuration bit of configurable event input 51(1)
0: Rising trigger disabled (for event and interrupt) for input line 51
1: Rising trigger enabled (for event and interrupt) for input line 51

Bit 18 Reserved, must be kept at reset value.

Bit 17 **RT49:** Rising trigger event configuration bit of configurable event input 49(1)
0: Rising trigger disabled (for event and interrupt) for input line 49
1: Rising trigger enabled (for event and interrupt) for input line 49

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **RT46:** Rising trigger event configuration bit of configurable event input 46(1)
0: Rising trigger disabled (for event and interrupt) for input line 46
1: Rising trigger enabled (for event and interrupt) for input line 46

Bits 13:3 Reserved, must be kept at reset value.

Bit 2 **RT34:** Rising trigger event configuration bit of configurable event input 34(1)
0: Rising trigger disabled (for event and interrupt) for input line 34
1: Rising trigger enabled (for event and interrupt) for input line 34

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

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs while the register is being written, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

16.6.5 EXTI falling trigger selection register 2 (EXTI_FTSR2)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FT63	FT62	FT61	FT60	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT51	Res.	FT49	Res.
rw	rw	rw	rw									rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FT46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT34	Res.	Res.
	rw												rw		

Bits 31:28 **FTi:** Falling trigger event configuration bit of configurable event input i (i = 63 to 60)(1)
0: Falling trigger disabled (for event and interrupt) for input line i
1: Falling trigger enabled (for event and interrupt) for input line i

Bits 27:20 Reserved, must be kept at reset value.

Bit 19 **FT51:** Falling trigger event configuration bit of configurable event input 51(1)
0: Falling trigger disabled (for event and interrupt) for input line 51
1: Falling trigger enabled (for event and interrupt) for input line 51

Bit 18 Reserved, must be kept at reset value.

Bit 17 **FT49:** Falling trigger event configuration bit of configurable event input 49(1)
0: Falling trigger disabled (for event and interrupt) for input line 49
1: Falling trigger enabled (for event and interrupt) for input line 49

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **FT46:** Falling trigger event configuration bit of configurable event input 46(1)
0: Falling trigger disabled (for event and interrupt) for input line 46
1: Falling trigger enabled (for event and interrupt) for input line 46

Bits 13:3 Reserved, must be kept at reset value.

Bit 2 **FT34:** Falling trigger event configuration bit of configurable event input 34(1)
0: Falling trigger disabled (for event and interrupt) for input line 34
1: Falling trigger enabled (for event and interrupt) for input line 34

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

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs while the register is being written, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

16.6.6 EXTI software interrupt event register 2 (EXTI_SWIER2)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SW63	SW62	SW61	SW60	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW51	Res.	SW49	Res.
rw	rw	rw	rw									rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SW46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW34	Res.	Res.
	rw												rw		

Bits 31:28 **SWi:** Software interrupt on event i (i = 63 to 60)
This bitfield always returns 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit triggers an event on line i. This bit is auto cleared by hardware.

Bits 27:20 Reserved, must be kept at reset value.

Bit 19 **SW51:** Software interrupt on event 51
This bitfield always returns 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit triggers an event on line 51. This bit is auto cleared by hardware.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **SW49:** Software interrupt on event 49
This bitfield always returns 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit triggers an event on line 49. This bit is auto cleared by hardware.

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **SW46:** Software interrupt on event 46
This bitfield always returns 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit triggers an event on line 46. This bit is auto cleared by hardware.

Bits 13:3 Reserved, must be kept at reset value.

Bit 2 **SW34:** Software interrupt on event 34
This bitfield always returns 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit triggers an event on line 34. This bit is auto cleared by hardware.

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

16.6.7 EXTI rising trigger selection register 3 (EXTI_RTSR3)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT71	RT70	RT69	RT68	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				

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

Bits 7:4 **RTi:** Rising trigger event configuration bit of configurable event input i (i = 71 to 68)(1)

0: Rising trigger disabled (for event and interrupt) for input line i

1: Rising trigger enabled (for event and interrupt) for input line i

Bits 3:0 Reserved, must be kept at reset value.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs while the register is being written, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

16.6.8 EXTI falling trigger selection register 3 (EXTI_FTSR3)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT71	FT70	FT69	FT68	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				

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

Bits 7:4 **FTi:** Falling trigger event configuration bit of configurable event input i (i = 71 to 68)(1)

0: Falling trigger disabled (for event and interrupt) for input line i

1: Falling trigger enabled (for event and interrupt) for input line i

Bits 3:0 Reserved, must be kept at reset value.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs while the register is being written, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

16.6.9 EXTI software interrupt event register 3 (EXTI_SWIER3)

Address offset: 0x48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW71	SW70	SW69	SW68	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				

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

Bits 7:4 **SWi:** Software interrupt on event i (i = 71 to 68)
Always returns 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit triggers an event on line i. This bit is auto cleared by HW.

Bits 3:0 Reserved, must be kept at reset value.

16.6.10 EXTI interrupt mask register 1 (EXTI_IMR1)

Address offset: 0x80

Reset value: 0xFFC0 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 **IMi:** CPU interrupt mask on event input i (i = 31 to 22)(1)
0: Interrupt request from line x is masked
1: Interrupt request from line x is unmasked

Bits 21:0 **IMi:** CPU interrupt mask on event input i (i = 21 to 0)(2)
0: Interrupt request from line x is masked
1: Interrupt request from line x is unmasked

1. The reset value for direct event inputs is set to 1 in order to enable the interrupt by default.
2. The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.

16.6.11 EXTI event mask register 1 (EXTI_EMR1)

Address offset: 0x84

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EM31	EM30	EM29	EM28	EM27	EM26	EM25	EM24	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 **EMi:** CPU *event* mask on event input i (i = 31 to 22)
0: Event request from line i is masked
1: Event request from line i is unmasked

Bits 21:0 **EMi:** CPU *event* mask on event input i (i = 21 to 0)
0: Event request from line i is masked
1: Event request from line i is unmasked

16.6.12 EXTI pending register 1 (EXTI_PR1)

Address offset: 0x88

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR21	PR20	PR19	PR18	PR17	PR16
										rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bits 21:0 **PRi:** Configurable event input i pending (i = 21 to 0)
0: No trigger request occurred
1: Selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

16.6.13 EXTI interrupt mask register 2 (EXTI_IMR2)

Address offset: 0x90

Reset value: 0x0FB5 BFFB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM63	IM62	IM61	IM60	IM59	IM58	IM57	IM56	IM55	Res.	IM53	IM52	IM51	IM50	IM49	IM48
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM47	IM46	IM45	IM44	IM43	IM42	IM41	IM40	IM39	IM38	IM37	IM36	IM35	IM34	IM33	IM32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 **IMi:** CPU interrupt mask on event input i (i = 63 to 55)[1][2]
0: Interrupt request from line i is masked
1: Interrupt request from line i is unmasked

Bit 22 Reserved, must be kept at reset value.

Bits 21:0 **IMi:** CPU interrupt mask on event input i (i = 53 to 32)[1][2]
0: Interrupt request from line i is masked
1: Interrupt request from line i is unmasked

1. The reset value for direct event inputs is set to 1 in order to enable the interrupt by default.
2. The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.

16.6.14 EXTI event mask register 2 (EXTI_EMR2)

Address offset: 0x94

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EM63	EM62	EM61	EM60	EM59	EM58	EM57	EM56	EM55	Res.	EM53	EM52	EM51	EM50	EM49	EM48
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM47	EM46	EM45	EM44	EM43	EM42	EM41	EM40	EM39	EM38	EM37	EM36	EM35	EM34	EM33	EM32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 **EMi:** CPU *event* mask on event input i (i = 63 to 55)
0: Event request from line i is masked
1: Event request from line i is unmasked

Bit 22 Reserved, must be kept at reset value.

Bits 21:0 **EMi:** CPU *event* mask on event input i (i = 53 to 32)
0: Event request from line i is masked
1: Event request from line i is unmasked

16.6.15 EXTI pending register 2 (EXTI_PR2)

Address offset: 0x98

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PR63	PR62	PR61	PR60	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR51	Res.	PR49	Res.
rc_w1	rc_w1	rc_w1	rc_w1									rc_w1		rc_w1	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PR46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR34	Res.	Res.
	rc_w1												rc_w1		

Bits 31:28 **PRi:** Configurable event input i pending (i = 63 to 60)
0: No trigger request occurred
1: Selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

Bits 27:20 Reserved, must be kept at reset value.

Bit 19 **PR51:** Configurable event input 51 pending bit
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PR49:** Configurable event input 49 pending bit
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **PR46:** Configurable event input 46 pending bit
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

Bits 13:3 Reserved, must be kept at reset value.

Bit 2 **PR34:** Configurable event input 34 pending bit
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

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

16.6.16 EXTI interrupt mask register 3 (EXTI_IMR3)

Address offset: 0xA0

Reset value: 0x0000 3F0F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	IM77	Res.	Res.	Res.	Res.	Res.	IM71	IM70	IM69	IM68	IM67	IM66	IM65	IM64
		rw						rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **IM77:** CPU interrupt mask on direct event input 77 (1)
0: Interrupt request from line 77 is masked
1: Interrupt request from line 77 is unmasked

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:0 **IMi:** CPU interrupt mask on direct event input i (i = 71 to 64)(1)(2)
0: Interrupt request from line i is masked
1: Interrupt request from line i is unmasked

1. The reset value for direct event inputs is set to 1 in order to enable the interrupt by default.
2. The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.

16.6.17 EXTI event mask register 3 (EXTI_EMR3)

Address offset: 0xA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	EM77	Res.	Res.	Res.	Res.	Res.	EM71	EM70	EM69	EM68	EM67	EM66	EM65	EM64
		rw						rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **EM77:** CPU *event* mask on event input 77
0: Event request from line 77 is masked
1: Event request from line 77 is unmasked

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:0 **EMi:** CPU *event* mask on event input i (i = 71 to 64)
0: Event request from line i is masked
1: Event request from line i is unmasked

16.6.18 EXTI pending register 3 (EXTI_PR3)

Address offset: 0xA8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR71	PR70	PR69	PR68	Res.	Res.	Res.	Res.
								rc_w1	rc_w1	rc_w1	rc_w1				

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

Bits 7:4 **PRi:** Configurable event input i pending bit (i = 71 to 68)

0: No trigger request occurred

1: selected trigger request occurred

This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

Bits 3:0 Reserved, must be kept at reset value.

16.6.19 EXTI register map

Table 114. Extended interrupt and event controller register map and reset values

0ffset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**EXTI_RTSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT21	RT20	RT19	RT18	RT17	RT16	RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**EXTI_FTSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT21	FT20	FT19	FT18	FT17	FT16	FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**EXTI_SWIER1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW21	SW20	SW19	SW18	SW17	SW16	SW15	SW14	SW13	SW12	SW11	SW10	SW9	SW8	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C-0x1C	Reserved	Res.																															
0x20	**EXTI_RTSR2**	RT63	RT62	RT61	RT60	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT51	Res.	RT49	Res.	Res.	RT46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT34	Res.	Res.
	Reset value	0	0	0	0									0		0			0												0		
0x24	**EXTI_FTSR2**	FT63	FT62	FT61	FT60	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT51	Res.	FT49	Res.	Res.	FT46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT34	Res.	Res.
	Reset value	0	0	0	0									0		0			0												0		
0x28	**EXTI_SWIER2**	SW63	SW62	SW61	SW60	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW51	Res.	SW49	Res.	Res.	SW46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW34	Res.	Res.
	Reset value	0	0	0	0									0		0			0												0		
0x2C-0x3C	Reserved	Res.																															
0x40	EXTI_RTSR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT71	RT70	RT69	RT68	Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				
0x44	EXTI_FTSR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT71	FT70	FT69	FT68	Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				
0x48	EXTI_SWIER3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW71	SW70	SW69	SW68	Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				
0x4C-0x7C	Reserved	Res.																															
0x80	**EXTI_IMR1**	IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16	IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
	Reset value	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	**EXTI_EMR1**	EM31	EM30	EM29	EM28	EM27	EM26	EM25	EM24	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16	EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	**EXTI_PR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR21	PR20	PR19	PR18	PR17	PR16	PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
	Reset value											X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x8C	Reserved																																

Table 114. Extended interrupt and event controller register map and reset values (continued)

0ffset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x90	**EXTI_IMR2**	IM63	IM62	IM61	IM60	IM59	IM58	IM57	IM56	IM55	Res.	IM53	IM52	IM51	IM50	IM49	IM48	IM47	IM46	IM45	IM44	IM43	IM42	IM41	IM40	IM39	IM38	IM37	IM36	IM35	IM34	IM33	IM32
	Reset value	0	0	0	0	1	1	1	1	1		1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
0x94	**EXTI_EMR2**	EM63	EM62	EM61	EM60	EM59	EM58	EM57	EM56	EM55	Res.	EM53	EM52	EM51	EM50	EM49	EM48	EM47	EM46	EM45	EM44	EM43	EM42	EM41	EM40	EM39	EM38	EM37	EM36	EM35	EM34	EM33	EM32
	Reset value	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98	**EXTI_PR2**	PR63	PR62	PR61	PR60	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR51	Res.	PR49	Res.	Res.	PR46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR34	Res.	Res.
	Reset value	X	X	X	X									X		X			X												X		
0x9C	Reserved																																
0xA0	**EXTI_IMR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM77	Res.	Res.	Res.	Res.	Res.	IM71	IM70	IM69	IM68	IM67	IM66	IM65	IM64
	Reset value																			1						0	0	0	0	1	1	1	1
0xA4	**EXTI_EMR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM77	Res.	Res.	Res.	Res.	Res.	EM71	EM70	EM69	EM68	EM67	EM66	EM65	EM64
	Reset value																			0						0	0	0	0	0	0	0	0
0xA8	**EXTI_PR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR71	PR70	PR69	PR68	Res.	Res.	Res.	Res.
	Reset value																									X	X	X	X				

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

17 Cyclic redundancy check calculation unit (CRC)

17.1 CRC introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

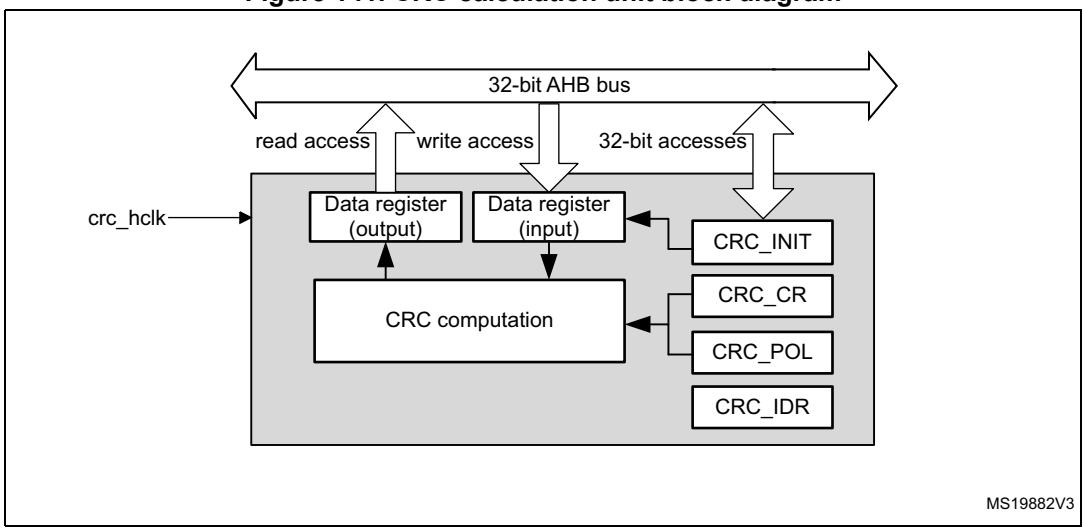
17.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-,16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility options on I/O data
- Accessed through AHB slave peripheral by 32-bit words only, with the exception of CRC_DR register that can be accessed by words, right-aligned half-words and right-aligned bytes

17.3 CRC functional description

17.3.1 CRC block diagram

Figure 141. CRC calculation unit block diagram



17.3.2 CRC internal signals

Table 115. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

17.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. When the RTYPE_IN bit is set in CRC_CR, non-word accesses to CRC_DR register are ignored. For the other registers only 32-bit accesses are allowed.

The duration of the computation depends on input data width:

- 4 AHB clock cycles for 32 bits
- 2 AHB clock cycles for 16 bits
- 1 AHB clock cycles for 8 bits

An input buffer allows a second data to be immediately written without waiting for any wait-states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register. If RTYPE_IN bit is set in this register, the input reversing operation can be performed with a byte or half-word granularity, but not with single bit granularity.

For example, with RTYPE_IN cleared, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x58D43CB2 with bit-reversal done by byte (REV_IN[1:0] = 01)
- 0xD458B23C with bit-reversal done by half-word (REV_IN[1:0] = 10)
- 0xB23CD458 with bit-reversal done on the full word (REV_IN[1:0] = 11)

With RTYPE_IN set, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x3C4D1A2B with half-word reversal done by word (REV_IN[1:0] = 01)
- 0x4D3C2B1A with byte-reversal done by word (REV_IN[1:0] = 10)

The output data can also be reversed by setting the REV_OUT[1:0] bits in the CRC_CR register. If RTYPE_OUT is set in this register, the output reversing operation is performed with a byte or half-word granularity, but not with single bit granularity.

With RTYPE_OUT cleared, the operation is done at bit level. For example, 0x11223344 output data are converted to 0x22CC4488.

With RTYPE_OUT set, 0x11223344 output data are converted as:

- 0x33441122 with half-word reversal done by word (REV_IN[1:0] = 01)
- 0x44332211 with byte-reversal done by word (REV_IN[1:0] = 10)

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC 7-, 8-, 16-, and 32-bit calculations. It is not affected by the RESET bit in the CRC_CR register.

Polynomial programmability

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7-, 8-, 16-, or 32-bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

Note: The type of an even polynomial is $X+X^2+..+X^n$, while the type of an odd polynomial is $1+X+X^2+..+X^n$.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

17.4 CRC registers

The CRC_DR register can be accessed by words, right-aligned half-words and right-aligned bytes when RTYPE bit is cleared in CRC_CR. When RTYPE is set, only word accesses are allowed. For the other registers only 32-bit accesses are allowed.

17.4.1 CRC data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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

Bits 31:0 **DR[31:0]:** Data register bits

This register is used to write new data to the CRC calculator.

It holds the previous CRC calculation result when it is read.

If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

17.4.2 CRC independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bits 31:0 **IDR[31:0]**: General-purpose 32-bit data register bits

These bits can be used as a temporary storage location for four bytes.

This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register

17.4.3 CRC control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **RTYPE_OUT**: Reverse type output
This bit controls the reversal granularity of the output data.
0: Bit level output
1: Byte or half-word level output

Bit 9 **RTYPE_IN**: Reverse type input
This bit controls the reversal granularity of the input data.
0: Bit level input
1: Byte or half-word level input

Bits 8:7 **REV_OUT[1:0]**: Reverse output data
This bitfield controls the reversal of the bit order of the output data.
00: Bit order not affected (RTYPE_OUT = 0 or 1)
01: Bit-reversed output format (RTYPE_OUT = 0) or half-word reversal done by word (RTYPE_OUT = 1)
10: Bit order not affected (RTYPE_OUT = 0) or byte reversal done by word (RTYPE_OUT = 1)
11: Bit order not affected (RTYPE_OUT = 0 or 1)

Bits 6:5 **REV_IN[1:0]**: Reverse input data
This bitfield controls the reversal of the bit order of the input data
00: Bit order not affected (RTYPE_IN = 0 or 1)
01: Bit reversal done by byte (RTYPE_IN = 0) or half-word reversal done by word (RTYPE_IN = 1)
10: Bit reversal done by half-word (RTYPE_IN = 0) or byte reversal done by word (RTYPE_IN = 1)
11: Bit reversal done by word (RTYPE_IN = 0) or bit order is not affected (RTYPE_IN = 1)

Bits 4:3 **POLYSIZE[1:0]**: Polynomial size

These bits control the size of the polynomial.

00: 32 bit polynomial

01: 16 bit polynomial

10: 8 bit polynomial

11: 7 bit polynomial

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RESET**: RESET bit

This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware.

17.4.4 CRC initial value (CRC_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

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

Bits 31:0 **CRC_INIT[31:0]**: Programmable initial CRC value

This register is used to write the CRC initial value.

17.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C1 1DB7

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

Bits 31:0 **POL[31:0]**: Programmable polynomial

This register is used to write the coefficients of the polynomial to be used for CRC calculation.

If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.

17.4.6 CRC register map

Table 116. CRC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRC_DR	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	CRC_IDR	IDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	CRC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTYPE_OUT	RTYPE_IN	REV_OUT[1:0]		REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
	Reset value																						0	0	0	0	0	0	0	0			0
0x10	CRC_INIT	CRC_INIT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	CRC_POL	POL[31:0]																															
	Reset value	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	1

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

18 CORDIC coprocessor (CORDIC)

18.1 CORDIC introduction

The CORDIC coprocessor provides hardware acceleration of mathematical functions (mainly trigonometric ones) commonly used in motor control, metering, signal processing, and many other applications.

It speeds up the calculation of these functions compared to a software implementation, making possible the use of a lower operating frequency, or freeing up processor cycles to perform other tasks.

18.2 CORDIC main features

- 24-bit CORDIC rotation engine
- Circular and Hyperbolic modes
- Rotation and Vectoring modes
- Functions: sine, cosine, sinh, cosh, atan, atanh, phase, modulus, square root, natural logarithm
- Programmable precision
- Low latency AHB slave interface
- Results can be read as soon as ready, without polling or interrupt
- DMA read and write channels
- Multiple register read/write by DMA

18.3 CORDIC functional description

18.3.1 General description

The CORDIC is a cost-efficient successive approximation algorithm for evaluating trigonometric and hyperbolic functions.

In trigonometric (circular) mode, the sine and cosine of an angle θ are determined by rotating the unit vector [1, 0] through decreasing angles until the cumulative sum of the rotation angles equals the input angle θ. The x and y cartesian components of the rotated vector then correspond, respectively, to the cosine and sine of θ. Inversely, the angle of a vector [x, y] corresponding to arctangent (y / x), is determined by rotating [x, y] through successively decreasing angles to obtain the unit vector [1, 0]. The cumulative sum of the rotation angles gives the angle of the original vector.

The CORDIC algorithm can also be used for calculating hyperbolic functions (sinh, cosh, atanh), by replacing the successive circular rotations by steps along a hyperbole.

Other functions can be derived from the basic functions described above.

18.3.2 CORDIC functions

The first step when using the coprocessor is to select the required function, by programming the FUNC field of the CORDIC_CR register.

Table 117 lists the functions supported by the CORDIC coprocessor.

Table 117. CORDIC functions

Function	Primary argument (ARG1)	Secondary argument (ARG2)	Primary result (RES1)	Secondary result (RES2)
Cosine	angle θ	modulus m	$m \cdot \cos\theta$	$m \cdot \sin\theta$
Sine	angle θ	modulus m	$m \cdot \sin\theta$	$m \cdot \cos\theta$
Phase	x	y	atan2(y,x)	$\sqrt{x^2+y^2}$
Modulus	x	y	$\sqrt{x^2+y^2}$	atan2(y,x)
Arctangent	x	none	atan x	none
Hyperbolic cosine	x	none	cosh x	sinh x
Hyperbolic sine	x	none	sinh x	cosh x
Hyperbolic arctangent	x	none	atanh x	none
Natural logarithm	x	none	ln x	none
Square root	x	none	$\sqrt{x}$	none

Several functions take two input arguments (ARG1 and ARG2), and some generate two results (RES1 and RES2) simultaneously. This is a side-effect of the algorithm and means that only one operation is needed to obtain two values. This is the case, for example, when performing polar-to-rectangular conversion: sin θ also generates cos θ, and cos θ also generates sin θ. Similarly for rectangular-to-polar conversion (phase(x,y), modulus(x,y)) and for hyperbolic functions (cosh θ, sinh θ).

Note: The exponential function, exp x, can be obtained as the sum of sinh x and cosh x. Furthermore, base N logarithms, $\log_N x$, can be derived by multiplying ln x by a constant K, where K = 1/ln N.

For certain functions (atan, log, sqrt) a scaling factor (see *Section 18.3.4*) can be applied to extend the range beyond the maximum [-1, 1] supported by the q1.31 fixed point format. The scaling factor must be set to 0 for all other circular functions, and to 1 for hyperbolic functions.

Cosine

Table 118. Cosine parameters

Parameter	Description	Range
ARG1	Angle θ in radians, divided by π	[-1, 1]
ARG2	Modulus m	[0, 1]
RES1	m • cos θ	[-1, 1]

Table 118. Cosine parameters (continued)

Parameter	Description	Range
RES2	m • sin θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the cosine of an angle in the range -π to π. It can also be used to perform polar to rectangular conversion.

The primary argument is the angle θ in radians. It must be divided by π before programming ARG1.

The secondary argument is the modulus m. If m is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the cosine of the angle, multiplied by the modulus.

The secondary result, RES2, is the sine of the angle, multiplied by the modulus.

Sine

Table 119. Sine parameters

Parameter	Description	Range
ARG1	Angle θ in radians, divided by π	[-1, 1]
ARG2	Modulus m	[0, 1]
RES1	m • sin θ	[-1, 1]
RES2	m • cos θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the sine of an angle in the range -π to π. It can also be used to perform polar to rectangular conversion.

The primary argument is the angle θ in radians. It must be divided by π before programming ARG1.

The secondary argument is the modulus m. If m is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the sine of the angle, multiplied by the modulus.

The secondary result, RES2, is the cosine of the angle, multiplied by the modulus.

Phase

Table 120. Phase parameters

Parameter	Description	Range
ARG1	x coordinate	[-1, 1]
ARG2	y coordinate	[-1, 1]
RES1	Phase angle θ in radians, divided by π	[-1, 1]

Table 120. Phase parameters (continued)

Parameter	Description	Range
RES2	Modulus m	[0, 1]
SCALE	Not applicable	0

This function calculates the phase angle in the range -π to π of a vector **v** = [x y] (also known as atan2(y,x). It can also be used to perform rectangular to polar conversion.

The primary argument is the x coordinate, that is, the magnitude of the vector in the direction of the x axis. If |x| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG1.

The secondary argument is the y coordinate, that is, the magnitude of the vector in the direction of the y axis. If |y| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the phase angle θ of the vector **v**. RES1 must be multiplied by π to obtain the angle in radians. Note that values close to π may sometimes wrap to -π due to the circular nature of the phase angle.

The secondary result, RES2, is the modulus, given by: $|v| = \sqrt{x^2 + y^2}$. If |**v**| > 1 the result in RES2 is saturated to 1.

Modulus

Table 121. Modulus parameters

Parameter	Description	Range
ARG1	x coordinate	[-1, 1]
ARG2	y coordinate	[-1, 1]
RES1	Modulus m	[0, 1]
RES2	Phase angle θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the magnitude, or modulus, of a vector **v** = [x y]. It can also be used to perform rectangular to polar conversion.

The primary argument is the x coordinate, that is, the magnitude of the vector in the direction of the x axis. If |x| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG1.

The secondary argument is the y coordinate, that is, the magnitude of the vector in the direction of the y axis. If |y| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the modulus, given by: $|v| = \sqrt{x^2 + y^2}$. If |**v**| > 1 the result in RES1 is saturated to 1.

The secondary result, RES2, is the phase angle θ of the vector **v**. RES2 must be multiplied by π to obtain the angle in radians. Note that values close to π may sometimes wrap to -π due to the circular nature of the phase angle.

Arctangent

Table 122. Arctangent parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-1, 1]
ARG2	Not applicable	-
RES1	2^{-n} • atan x, in radians, divided by p	[-1, 1]
RES2	Not applicable	-
SCALE	n	[0 7]

This function calculates the arctangent, or inverse tangent, of the input argument x.

The primary argument, ARG1, is the input value, x = tan θ. If |x| > 1, a scaling factor of 2^{-n} must be applied in software such that $-1 < x \cdot 2^{-n} < 1$. The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the scale factor n must be programmed in the SCALE parameter.

Note that the maximum input value allowed is tan θ = 128, which corresponds to an angle θ = 89.55 degrees. For |x| > 128, a software method must be used to find atan x.

The secondary argument, ARG2, is unused.

The primary result, RES1, is the angle θ = atan x. RES1 must be multiplied by $2^n \cdot \pi$ to obtain the angle in radians.

The secondary result, RES2, is unused.

Hyperbolic cosine

Table 123. Hyperbolic cosine parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.559 0.559]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \cosh x$	[0.5 0.846]
RES2	$2^{-n} \cdot \sinh x$	[-0.683 0.683]
SCALE	n	1

This function calculates the hyperbolic cosine of a hyperbolic angle x. It can also be used to calculate the exponential functions $e^x = \cosh x + \sinh x$, and $e^{-x} = \cosh x - \sinh x$.

The primary argument is the hyperbolic angle x. Only values of x in the range -1.118 to +1.118 are supported. Since the minimum value of cosh x is 1, which is beyond the range of the q1.31 format, a scaling factor of 2^{-n} must be applied in software. The factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result, RES1, is the hyperbolic cosine, cosh x. RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic sine, sinh x. RES2 must be multiplied by 2 to obtain the correct result.

Hyperbolic sine

Table 124. Hyperbolic sine parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.559, 0.559]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \sinh x$	[-0.683, 0.683]
RES2	$2^{-n} \cdot \cosh x$	[0.5, 0.846]
SCALE	n	1

This function calculates the hyperbolic sine of a hyperbolic angle x. It can also be used to calculate the exponential functions e^x = cosh x + sinh x, and e^{-x} = cosh x- sinh x.

The primary argument is the hyperbolic angle x. Only values of x in the range -1.118 to +1.118 are supported. For all input values, a scaling factor of 2^{-n} must be applied in software, where n = 1. The scaled value x • 0.5 is programmed in ARG1 and the factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result, RES1, is the hyperbolic sine, sinh x. RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic cosine, cosh x. RES2 must be multiplied by 2 to obtain the correct result.

Hyperbolic arctangent

Table 125. Hyperbolic arctangent parameters

Parameter	Description	Range
ARG1	$x \bullet 2^{-n}$	[-0.403 0.403]
ARG2	Not applicable	-
RES1	$2^{-n} \bullet$atanh x	[-0.559 0.559]
RES2	Not applicable	-
SCALE	n	1

This function calculates the hyperbolic arctangent of the input argument x.

The primary argument is the input value x. Only values of x in the -0.806 to +0.806 range are supported. The value x must be scaled by a factor 2^{-n}, where n = 1. The scaled value

x • 0.5 is programmed in ARG1 and the factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result is the hyperbolic arctangent, atanh x. RES1 must be multiplied by 2 to obtain the correct value.

The secondary result is not used.

Natural logarithm

Table 126. Natural logarithm parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[0.054 0.875]
ARG2	Not applicable	-
RES1	$2^{-(n+1)}.\ln x$	[-0.279 0.137]
RES2	Not applicable	-
SCALE	n	[1 4]

This function calculates the natural logarithm of the input argument x.

The primary argument is the input value x. Only values of x in the range 0.107 to 9.35 are supported. The value x must be scaled by a factor 2^{-n}, such that $x < 2^{-n} < 1-2^{-n}$. The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the factor n must be programmed in the SCALE parameter.

Table 127 lists the valid scaling factors, n, and the corresponding ranges of x and ARG1.

Table 127. Natural log scaling factors and corresponding ranges

n	x range	ARG1 range
1	$0.107 \le x < 1$	$0.0535 \le ARG1 < 0.5$
2	$1 \le x < 3$	$0.25 \le ARG1 < 0.75$
3	$3 \le x < 7$	$0.375 \le ARG1 < 0.875$
4	$7 \le x \le 9.35$	$0.4375 \le ARG1 < 0.584$

The secondary argument is not used.

The primary result is the natural logarithm, ln x. RES1 must be multiplied by $2^{(n+1)}$ to obtain the correct value.

The secondary result is not used.

Square root

Table 128. Square root parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[0.027 0.875]
ARG2	Not applicable	-
RES1	$2^{-n}\sqrt{x}$	[0.04 1]
RES2	Not applicable	-
SCALE	n	[0 2]

This function calculates the square root of the input argument x.

The primary argument is the input value x. Only values of x in the range 0.027 to 2.34 are supported. The value x must be scaled by a factor 2^{-n}, such that $x \cdot 2^{-n} < (1 - 2^{(-n-2)})$.

The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the factor n must be programmed in the SCALE parameter.

Table 129 lists the valid scaling factors, n, and the corresponding ranges of x and ARG1.

Table 129. Square root scaling factors and corresponding ranges

n	x range	ARG1 range
0	$0.027 \leq x < 0.75$	$0.027 \leq ARG1 < 0.75$
1	$0.75 \leq x < 1.75$	$0.375 \leq ARG1 < 0.875$
2	$1.75 \leq x \leq 2.341$	$0.4375 \leq ARG1 \leq 0.585$

The secondary argument is not used.

The primary result is the square root of x. RES1 must be multiplied by 2^n to obtain the correct value.

The secondary result is not used.

18.3.3 Fixed point representation

The CORDIC operates in fixed point signed integer format. Input and output values can be either q1.31 or q1.15.

In q1.31 format, numbers are represented by one sign bit and 31 fractional bits (binary decimal places). The numeric range is therefore -1 (0x80000000) to $1 - 2^{-31}$ (0x7FFFFFFF).

In q1.15 format, the numeric range is 1 (0x8000) to $1 - 2^{-15}$ (0x7FFF). This format has the advantage that two input arguments can be packed into a single 32-bit write, and two results can be fetched in one 32-bit read.

18.3.4 Scaling factor

Several of the functions listed in *Section 18.3.2* specify a scaling factor, SCALE. This allows the function input range to be extended to cover the full range of values supported by the CORDIC, without saturating the input, output, or internal registers. If the scaling factor is

required, it must be calculated by software and programmed into the SCALE field of the CORDIC_CSR register. The input arguments must be scaled accordingly before programming the scaled values in the CORDIC_WDATA register. The scaling must also be undone on the results read from the CORDIC_RDATA register.

Note: The scaling factor entails a loss of precision due to truncation of the scaled value.

18.3.5 Precision

The precision of the result is dependent on the number of CORDIC iterations. The algorithm converges at a constant rate of one binary digit per iteration for trigonometric functions (sine, cosine, phase, modulus), see *Figure 142*.

For hyperbolic functions (hyperbolic sine, hyperbolic cosine, natural logarithm), the convergence rate is less constant due to the peculiarities of the CORDIC algorithm (see *Figure 143*). The square root function converges at roughly twice the speed of the hyperbolic functions (see *Figure 144*).

Figure 142. CORDIC convergence for trigonometric functions

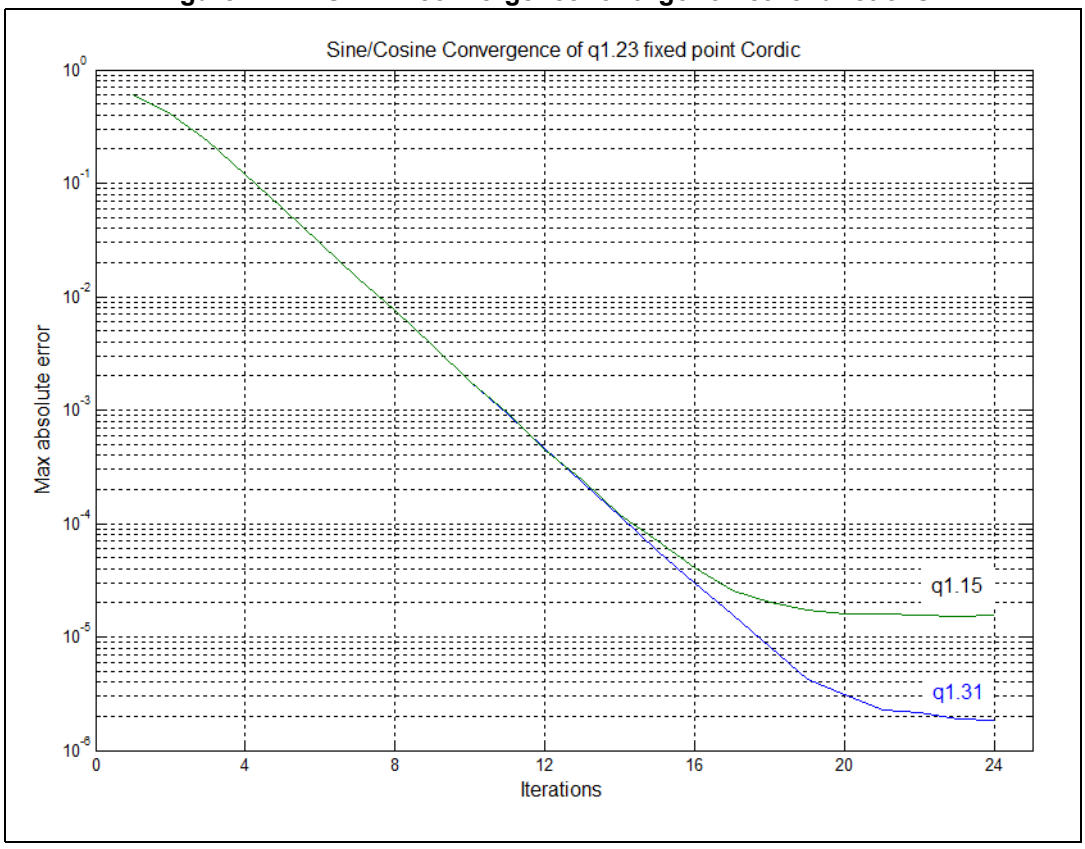


Figure 143. CORDIC convergence for hyperbolic functions

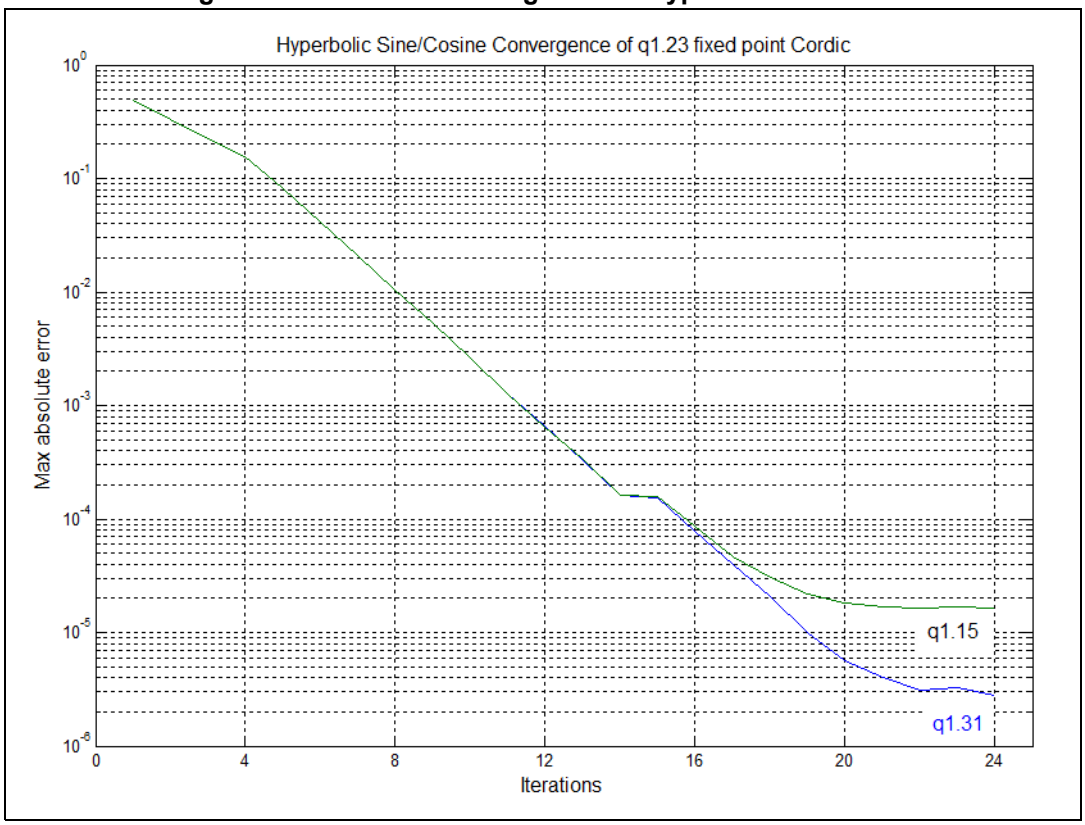
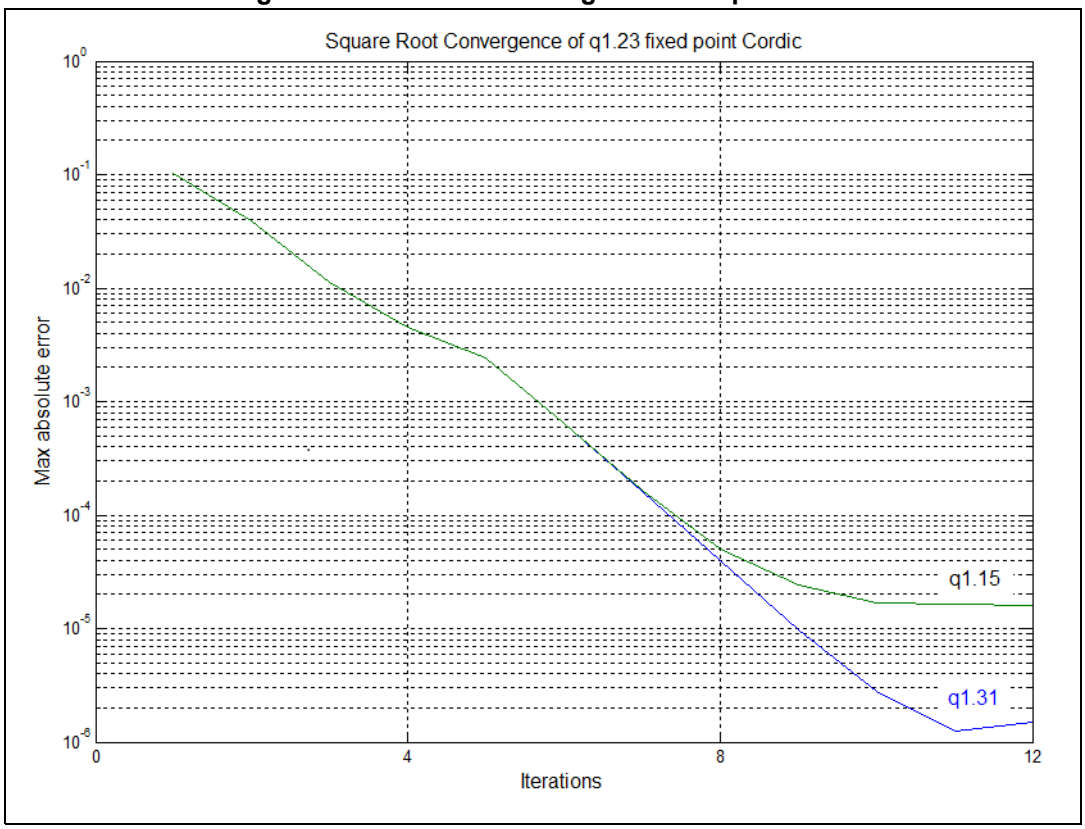


Figure 144. CORDIC convergence for square root



Note: The convergence rate decreases as the quantization error starts to become significant.

The CORDIC can perform four iterations per clock cycle. For each function, the maximum error remaining after every four iterations is shown in *Table 130*, together with the number of clock cycles required to reach that precision. From this table, the desired number of cycles can be determined and programmed in the PRECISION field of the CORDIC_CR register. The coprocessor stops as soon as the programmed number of iterations is completed, and the result can be read immediately.

Table 130. Precision vs. number of iterations

Function	Number of iterations	Number of cycles	Max residual error(1)	
			q1.31 format	q1.15 format
Sin, Cos, Phase(2), Mod, Atan(4)	4	1	2^{-3}	2^{-3}
	8	2	2^{-7}	2^{-7}
	12	3	2^{-11}	2^{-11}
	16	4	2^{-15}	2^{-15}
	20	5	2^{-18}	2^{-16}
	24	6	2^{-19}	2^{-16}

Table 130. Precision vs. number of iterations (continued)

Function	Number of iterations	Number of cycles	Max residual error(1)	
			q1.31 format	q1.15 format
Sinh, Cosh, Atanh, Ln(3)	4	1	2^{-2}	2^{-2}
	8	2	2^{-6}	2^{-6}
	12	3	2^{-10}	2^{-10}
	16	4	2^{-13}	2^{-13}
	20	5	2^{-17}	2^{-15}
	24	6	2^{-18}	2^{-15}
Sqrt(4)	4	1	2^{-7}	2^{-7}
	8	2	2^{-14}	2^{-14}
	12	3	2^{-19}	2^{-15}

1. Max residual error is the maximum error remaining after the given number of iterations, compared to the identical calculation performed in double precision floating point. An additional rounding error may be incurred, of up to 2^{-16} for q15 format or 2^{-20} for q31 format.
2. For modulus > 0.5. The achievable precision reduces proportionally to the magnitude of the modulus, as quantization error becomes significant.
3. SCALE = 1. If a higher scaling factor is used, the achievable precision is reduced proportionally.
4. SCALE = 0. If a higher scaling factor is used, the achievable precision is reduced proportionally.

18.3.6 Zero-overhead mode

The fastest way to use the coprocessor is to preprogram the CORDIC_CSR register with the function to be performed (FUNC), the desired number of clock cycles (PRECISION), the size of the input and output values (ARGSIZE, RESSIZE), the number of input arguments (NARGS) and/or results (NRES), and the scaling factor (SCALE), if applicable.

The calculation is triggered by writing the input arguments to the CORDIC_WDATA register. As soon as the correct number of input arguments has been written (and any ongoing calculation has finished), a new calculation is launched using these input arguments and the current CORDIC_CSR settings. There is no need to reprogram the CORDIC_CSR register if there is no change.

If a dual 32-bit input argument is needed (ARGSIZE = 0, NARGS = 1), the primary input argument (ARG1) must be written first, followed by the secondary argument (ARG2). If the secondary argument remains unchanged for a series of calculations, the second write can be avoided, by reprogramming the number of arguments to one (NARGS = 0), once the first calculation has started. The secondary argument retains its programmed value as long as the function is not changed.

Note: ARG2 is set to +1 (0x7FFFFFFF) after a reset.

If two 16-bit arguments are used (ARGSIZE = 1) they must be packed into a 32-bit word, with ARG1 in the least significant half-word and ARG2 in the most significant half-word. The packed 32-bit word is then written to the CORDIC_WDATA register. Only one write is needed in this case (NARGS = 0).

For functions taking only one input argument, ARG1, it is recommended to set NARGS = 0. If NARGS = 1, a second write to CORDIC_WDATA must be performed to trigger the calculation. The ARG2 data in this case is not used.

Once the calculation starts, any attempt to read the CORDIC_RDATA register inserts bus wait-states until the calculation is completed, before returning the result. It is then possible for the software to write the input and immediately read the result without polling to see if it is valid. Alternatively, the processor can wait for the appropriate number of clock cycles before reading the result. This time can be used to program the CORDIC_CSR register for the next calculation, and prepare the next input data, if needed. The CORDIC_CSR register can be reprogrammed while a calculation is in progress, without affecting the result of the ongoing calculation. In the same way, the CORDIC_WDATA register can be updated with the next argument(s) once the previous ones have been taken into account. The next arguments and settings remain pending until the previous calculation has completed.

When a calculation is finished, the result(s) can be read from the CORDIC_RDATA register. If two 32-bit results are expected (NRES = 1, RESSIZE = 0), the primary result (RES1) is read out first, followed by the secondary result (RES2). If only one 32-bit result is expected (NRES = 0, RESSIZE = 0), then RES1 is output on the first read.

If 16-bit results are expected (RESSIZE = 1), a single read to CORDIC_RDATA fetches both results packed into a 32-bit word. RES1 is in the lower half-word, and RES2 in the upper half-word. In this case, it is recommended to program NRES = 0. IF NRES = 1, a second read of CORDIC_RDATA must be performed to free up the CORDIC for the next operation. The data from this second read must be discarded.

The next calculation starts when the expected number of results has been read, provided the expected number of arguments has been written. This means that at any time, there can be a calculation in progress, or waiting for the results to be read, and an operation pending. Any further access to CORDIC_WDATA while an operation is pending cancels it and overwrites the data.

The following sequence summarizes the use of the CORDIC_IP in zero-overhead mode:

1. Program the CORDIC_CSR register with the appropriate settings
2. Program the argument(s) for the first calculation in the CORDIC_WDATA register. This launches the first calculation.
3. If needed, update the CORDIC_CSR register settings for the next calculation.
4. Program the argument(s) for the next calculation in the CORDIC_WDATA register.
5. Read the result(s) from the CORDIC_RDATA register. This triggers the next calculation.
6. Go to step *3*.

18.3.7 Polling mode

When a new result is available in the CORDIC_RDATA register, the RRDY flag is set in the CORDIC_CSR register. The flag can be polled by reading the register. It is reset by reading the CORDIC_RDATA register (once or twice, depending on the NRES field of the CORDIC_CSR register).

Polling the RRDY flag takes slightly longer than reading the CORDIC_RDATA register directly, since the result is not read as soon as it is available. The processor and bus interface are not stalled while reading the CORDIC_CSR register, so this mode may be of interest if stalling the processor is not acceptable (for example, if low latency interrupts must be serviced).

18.3.8 Interrupt mode

By setting the interrupt enable (IE) bit in the CORDIC_CSR register, an interrupt is generated whenever the RRDY flag is set. The interrupt is cleared when the flag is reset.

This mode allows the result of the calculation to be read under interrupt service routine, and hence given a priority relative to other tasks. However, it is slower than directly reading the result, or polling the flag, due to the interrupt handling delays.

18.3.9 DMA mode

If the DMA write enable (DMAWEN) bit is set in the CORDIC_CSR register, and no operation is pending, a DMA write channel request is generated. The DMA controller can transfer a primary input argument (ARG1) from memory into the CORDIC_WDATA register. Writing into the register deasserts the DMA request. If NARGS = 1 in the CORDIC_CSR register, a second DMA write channel request is generated to transfer the secondary input argument (ARG2) into the CORDIC_WDATA register. When all input arguments have been written, and any ongoing calculation has been completed (by reading the results), a new calculation is started and another DMA write channel request is generated.

If the DMA read enable (DMAREN) bit is set in the CORDIC_CSR register, the RRDY flag going active generates a DMA read channel request. The DMA controller can then transfer the primary result (RES1) from the CORDIC_RDATA register to memory. Reading the register deasserts the DMA request. If NRES = 1 in the CORDIC_CSR register, a second DMA request is generated to read out the secondary result (RES2). When all results have been read, the RRDY flag is deasserted.

The DMA read and write channels can be enabled separately. If both channels are enabled, the CORDIC can autonomously perform repeated calculations on a buffer of data without processor intervention. This allows the processor to perform other tasks. The DMA controller is operating in memory-to-peripheral mode for the write channel, and peripheral-to-memory mode for the read channel. The sequence is started by the processor setting the DMAWEN flag, the DMA read and write requests are generated as fast as the CORDIC can process the data.

In some cases, the input data may be stored in memory, and the output is transferred at regular intervals to another peripheral, such as a digital-to-analog converter. In this case, the destination peripheral generates a DMA request each time it needs a new data. The DMA controller can directly fetch the next sample from the CORDIC_RDATA register (in this case the DMA controller is operating in memory-to-peripheral mode, even though the source is a peripheral register). The act of reading the result allows the CORDIC to start a new calculation, which in turn generates a DMA write channel request, and the DMA controller transfers the next input value to the CORDIC_WDATA register. The DMA write channel is enabled (DMAWEN = 1), but the read channel must not be enabled.

In a similar way, data coming from another peripheral, such as an ADC, can be transferred directly to the CORDIC_WDATA register (in peripheral-to-memory mode). The DMA write channel must not be enabled. The CORDIC processes the input data and generates a DMA read request when complete, if DMAREN = 1. The DMA controller then transfers the result from CORDIC_RDATA register to memory (peripheral-to-memory mode).

Note: No DMA request is generated to program the CORDIC_CSR register. DMA mode is therefore useful only when repeatedly performing the same function with the same settings. The scale factor cannot be changed during a series of DMA transfers.

Note: Each DMA request must be acknowledged, as a result of the DMA performing an access to the CORDIC_WDATA or CORDIC_RDATA register. If an extraneous access to the relevant register occurs before this, the acknowledge is asserted prematurely, and may block the DMA channel. Therefore, when the DMA read channel is enabled, CPU access to the CORDIC_RDATA register must be avoided. Similarly, the processor must avoid accessing the CORDIC_WDATA register when the DMA write channel is enabled.

18.4 CORDIC registers

The CORDIC registers can be accessed only in 32-bit word format

18.4.1 CORDIC control/status register (CORDIC_CSR)

Address offset: 0x00

Reset value: 0x0000 0050

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARG SIZE	RES SIZE	N ARGS	NRES	DMA WEN	DMA REN	IEN
r									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SCALE[2:0]			PRECISION[3:0]				FUNC[3:0]			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RRDY**: Result ready flag

0: No new data in output register

1: CORDIC_RDATA register contains new data.

This bit is set by hardware when a CORDIC operation completes. It is reset by hardware when the CORDIC_RDATA register is read (NRES+1) times.

When this bit is set, if the IEN bit is also set, the CORDIC interrupt is asserted. If the DMAREN bit is set, a DMA read channel request is generated. While this bit is set, no new calculation is started.

Bits 30:23 Reserved, must be kept at reset value.

Bit 22 **ARGSIZE**: Width of input data

0: 32-bit

1: 16-bit

ARGSIZE selects the number of bits used to represent input data.

If 32-bit data is selected, the CORDIC_WDATA register expects arguments in q1.31 format.

If 16-bit data is selected, the CORDIC_WDATA register expects arguments in q1.15 format. The primary argument (ARG1) is written to the least significant half-word, and the secondary argument (ARG2) to the most significant half-word.

Bit 21 **RESSIZE**: Width of output data

0: 32-bit

1: 16-bit

RESSIZE selects the number of bits used to represent output data.

If 32-bit data is selected, the CORDIC_RDATA register contains results in q1.31 format.

If 16-bit data is selected, the least significant half-word of CORDIC_RDATA contains the primary result (RES1) in q1.15 format, and the most significant half-word contains the secondary result (RES2), also in q1.15 format.

Bit 20 **NARGS**: Number of arguments expected by the CORDIC_WDATA register

0: Only one 32-bit write (or two 16-bit values if ARGSIZE = 1) is needed for the next calculation.

1: Two 32-bit values must be written to the CORDIC_WDATA register to trigger the next calculation.

Reads return the current state of the bit.

Bit 19 **NRES**: Number of results in the CORDIC_RDATA register

0: Only one 32-bit value (or two 16-bit values if RESSIZE = 1) is transferred to the CORDIC_RDATA register on completion of the next calculation. One read from CORDIC_RDATA resets the RRDY flag.

1: Two 32-bit values are transferred to the CORDIC_RDATA register on completion of the next calculation. Two reads from CORDIC_RDATA are necessary to reset the RRDY flag.

Reads return the current state of the bit.

Bit 18 **DMAWEN**: Enable DMA write channel

0: Disabled. No DMA write requests are generated.

1: Enabled. Requests are generated on the DMA write channel whenever no operation is pending

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 17 **DMAREN**: Enable DMA read channel

0: Disabled. No DMA read requests are generated.

1: Enabled. Requests are generated on the DMA read channel whenever the RRDY flag is set.

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 16 **IEN**: Enable interrupt.

0: Disabled. No interrupt requests are generated.

1: Enabled. An interrupt request is generated whenever the RRDY flag is set.

This bit is set and cleared by software. A read returns the current state of the bit.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **SCALE[2:0]**: Scaling factor

The value of this field indicates the scaling factor applied to the arguments and/or results. A value n implies that the arguments have been multiplied by a factor 2^{-n}, and/or the results need to be multiplied by 2^{n}. Refer to *Section 18.3.2* for the applicability of the scaling factor for each function and the appropriate range.

Bits 7:4 **PRECISION[3:0]**: Precision required (number of iterations)

0: reserved

1 to 15: (Number of iterations)/4

To determine the number of iterations needed for a given accuracy refer to *Table 130*.

Note that for most functions, the recommended range for this field is 3 to 6.

Bits 3:0 **FUNC[3:0]**: Function

0: Cosine
1: Sine
2: Phase
3: Modulus
4: Arctangent
5: Hyperbolic cosine
6: Hyperbolic sine
7: Arctanh
8: Natural logarithm
9: Square root
10 to 15: Reserved

18.4.2 CORDIC argument register (CORDIC_WDATA)

Address offset: 0x04

Reset value: 0xXXXX XXXX

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

Bits 31:0 **ARG[31:0]**: Function input arguments

This register is programmed with the input arguments for the function selected in the CORDIC_CSR register FUNC field.

If 32-bit format is selected (CORDIC_CSR.ARGSIZE = 0) and two input arguments are required (CORDIC_CSR.NARGS = 1), two successive writes are required to this register. The first writes the primary argument (ARG1), the second writes the secondary argument (ARG2).

If 32-bit format is selected and only one input argument is required (NARGS = 0), only one write is required to this register, containing the primary argument (ARG1).

If 16-bit format is selected (CORDIC_CSR.ARGSIZE = 1), one write to this register contains both arguments. The primary argument (ARG1) is in the lower half, ARG[15:0], and the secondary argument (ARG2) is in the upper half, ARG[31:16]. In this case, NARGS must be set to 0.

Refer to *Section 18.3.2* for the arguments required by each function, and their permitted range.

When the required number of arguments has been written, the CORDIC evaluates the function designated by CORDIC_CSR.FUNC using the supplied input arguments, provided any previous calculation has completed. If a calculation is ongoing, the ARG1 and ARG 2 values are held pending until the calculation is completed and the results read. During this time, a write to the register cancels the pending operation and overwrite the argument data.

18.4.3 CORDIC result register (CORDIC_RDATA)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bits 31:0 **RES[31:0]**: Function result

If 32-bit format is selected (CORDIC_CSR.RESSIZE = 0) and two output values are expected (CORDIC_CSR.NRES = 1), this register must be read twice when the RRDY flag is set. The first read fetches the primary result (RES1). The second read fetches the secondary result (RES2) and resets RRDY.

If 32-bit format is selected and only one output value is expected (NRES = 0), only one read of this register is required to fetch the primary result (RES1) and reset the RRDY flag.

If 16-bit format is selected (CORDIC_CSR.RESSIZE = 1), this register contains the primary result (RES1) in the lower half, RES[15:0], and the secondary result (RES2) in the upper half, RES[31:16]. In this case, NRES must be set to 0, and only one read performed.

A read from this register resets the RRDY flag in the CORDIC_CSR register.

18.4.4 CORDIC register map

Table 131. CORDIC register map and reset value

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CORDIC_CSR	RRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARGSIZE	RESSIZE	NARGS	NRES	DMAWEN	DMAREN	IEN	Res.	Res.	Res.	Res.	SCALE [2:0]				PRECISION [3:0]				FUNC [3:0]			
	Reset value	0									0	0	0	0	0	0	0					0	0	0	0	0	1	0	1	0	0	0	0
0x04	CORDIC_WDATA	ARG[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x08	CORDIC_RDATA	RES[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

19 Filter math accelerator (FMAC)

19.1 FMAC introduction

The filter math accelerator unit performs arithmetic operations on vectors. It comprises a multiplier/accumulator (MAC) unit, together with address generation logic which allows it to index vector elements held in local memory.

The unit includes support for circular buffers on input and output, which allows digital filters to be implemented. Both finite and infinite impulse response filters can be realized.

The unit allows frequent or lengthy filtering operations to be offloaded from the CPU, freeing up the processor for other tasks. In many cases it can accelerate such calculations compared to a software implementation, resulting in a speed-up of time critical tasks.

19.2 FMAC main features

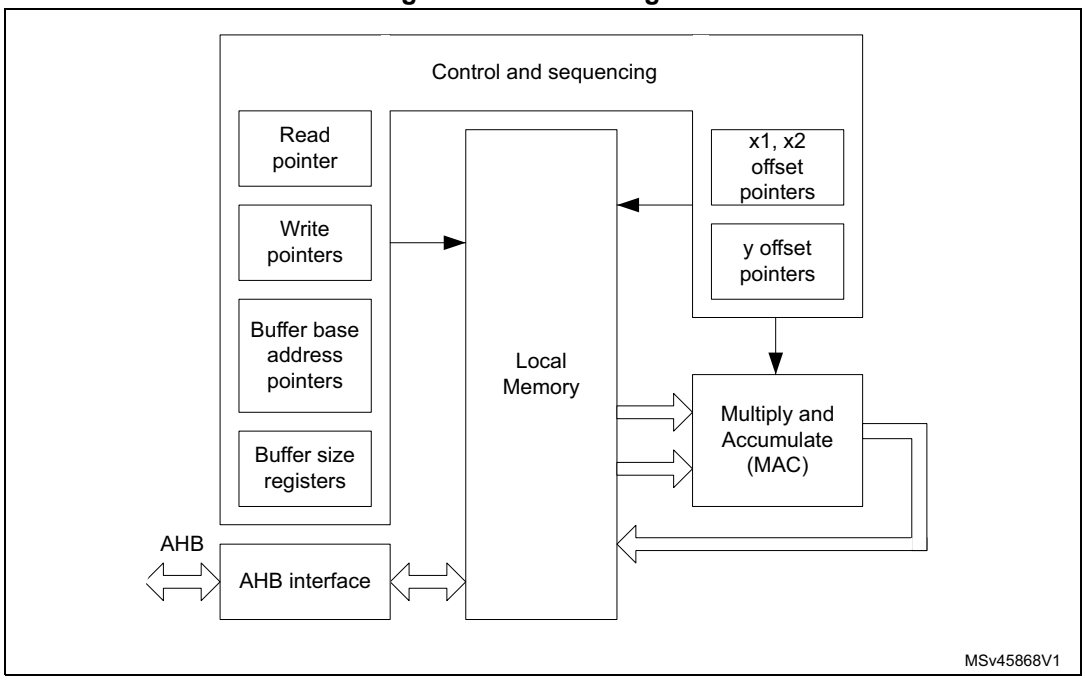
- 16 x 16-bit multiplier
- 24 + 2-bit accumulator with addition and subtraction
- 16-bit input and output data
- 256 x 16-bit local memory
- Up to three areas can be defined in memory for data buffers (two input, one output), defined by programmable base address pointers and associated size registers
- Input and output buffers can be circular
- Filter functions: FIR, IIR (direct form 1)
- Vector functions: Dot product, convolution, correlation
- AHB slave interface
- DMA read and write data channels

19.3 FMAC functional description

19.3.1 General description

The FMAC is shown in *Figure 145*.

Figure 145. Block diagram



The unit is built around a fixed point multiplier and accumulator (MAC). The MAC can take two 16-bit input signed values from memory, multiply them together and add them to the contents of the accumulator. The address of the input values in memory is determined using a set of pointers. These pointers can be loaded, incremented, decremented or reset by the internal hardware. The pointer and MAC operations are controlled by a built-in sequencer in order to execute the requested operation.

To calculate a dot product, the two input vectors are loaded into the local memory by the processor or DMA controller, and the requested operation is selected and started. Each pair of input vector elements is fetched from memory, multiplied together and accumulated. When all the vector elements have been processed, the contents of the accumulator are stored in the local memory, from where they can be read out by the processor or DMA.

The finite impulse response (FIR) filter operation (also known as convolution) consists in repeatedly calculating the dot product of the coefficient vector and a vector of input samples, the latter being shifted by one sample delay, with the least recent sample being discarded and a new sample added, at each repetition.

The infinite impulse response (IIR) filter operation is the convolution of the feedback coefficients with the previous output samples, added to the result of the FIR convolution.

A more detailed description of the filter operations is given in *Section 19.3.6: Filter functions*.

19.3.2 Local memory and buffers

The unit contains a 256 x 16-bit read/write memory which is used for local storage:

- Input values (the elements of the input vectors) are stored in two buffers, X1 and X2.
- Output values (the results of the operations) are stored in another buffer, Y.
- The locations and sizes of the buffers are designated as follows:
  - x1_base: the base address of the X1 buffer
  - x2_base: the base address of the X2 buffer
  - y_base: the base address of the Y buffer
  - x1_buf_size: the number of 16-bit addresses allocated to the X1 buffer
  - x2_buf_size: the number of 16-bit addresses allocated to the X2 buffer
  - y_buf_size: the number of 16-bit addresses allocated to the Y buffer.

These parameters are programmed in the corresponding registers when configuring the unit.

The CPU (or DMA controller) can initialize the contents of each buffer using the Initialization functions (*Section 19.3.5: Initialization functions*) and writing to the write data register. The data is transferred to the location within the target buffer indicated by a write pointer. After each new write, the write pointer is incremented. When the write pointer reaches the end of the allocated buffer space, it wraps back to the base address. This feature is used to load the elements of a vector prior to an operation, or to initialize a filter and load filter coefficients.

Buffer configuration

The buffer sizes and base address offsets must be configured in the X1, X2 and Y buffer configuration registers. For each function, the required buffer size is specified in the function description in *Section 19.3.6: Filter functions*. The base addresses can be chosen anywhere in internal memory, provided that all buffers fit within the internal memory address range (0x00 to 0xFF), that is, base address + buffer size must be less than 256.

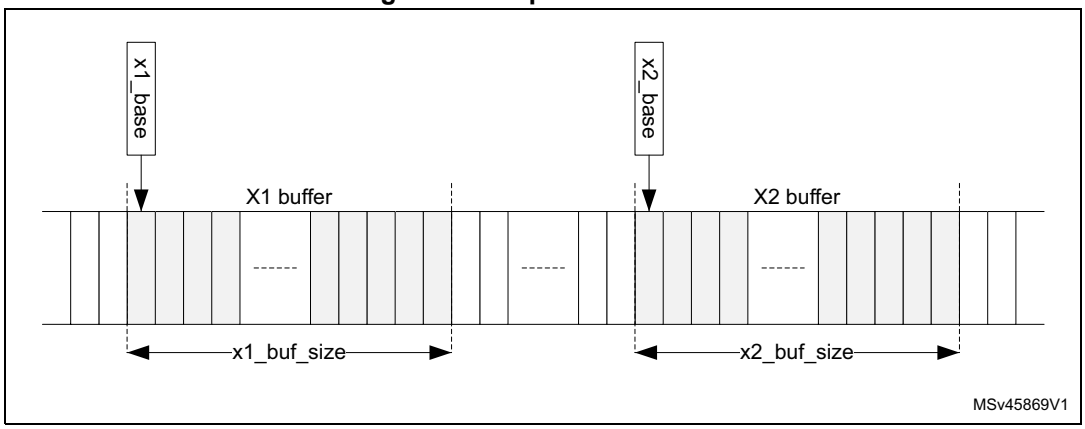
There is no constraint on the size and location of the buffers (they can overlap or even coincide exactly). For filter functions it is recommended not to overlap buffers as this can lead to erroneous behavior.

When circular buffer operation is required, an optional “headroom”, d, can be added to the buffer size. Furthermore, a watermark level can be set, to regulate the CPU or DMA activity. The value of d and the watermark level must be chosen according to the application performance requirements. For maximum throughput, the input buffer must never go empty, so d must be somewhat greater than the watermark level, allowing for any interrupt or DMA latency. On the other hand, if the input data can not be provided as fast as the unit can process them, the buffer can be allowed to empty waiting for the next data to be written, so d can be equal to the watermark level (to ensure that no overflow occurs on the input).

19.3.3 Input buffers

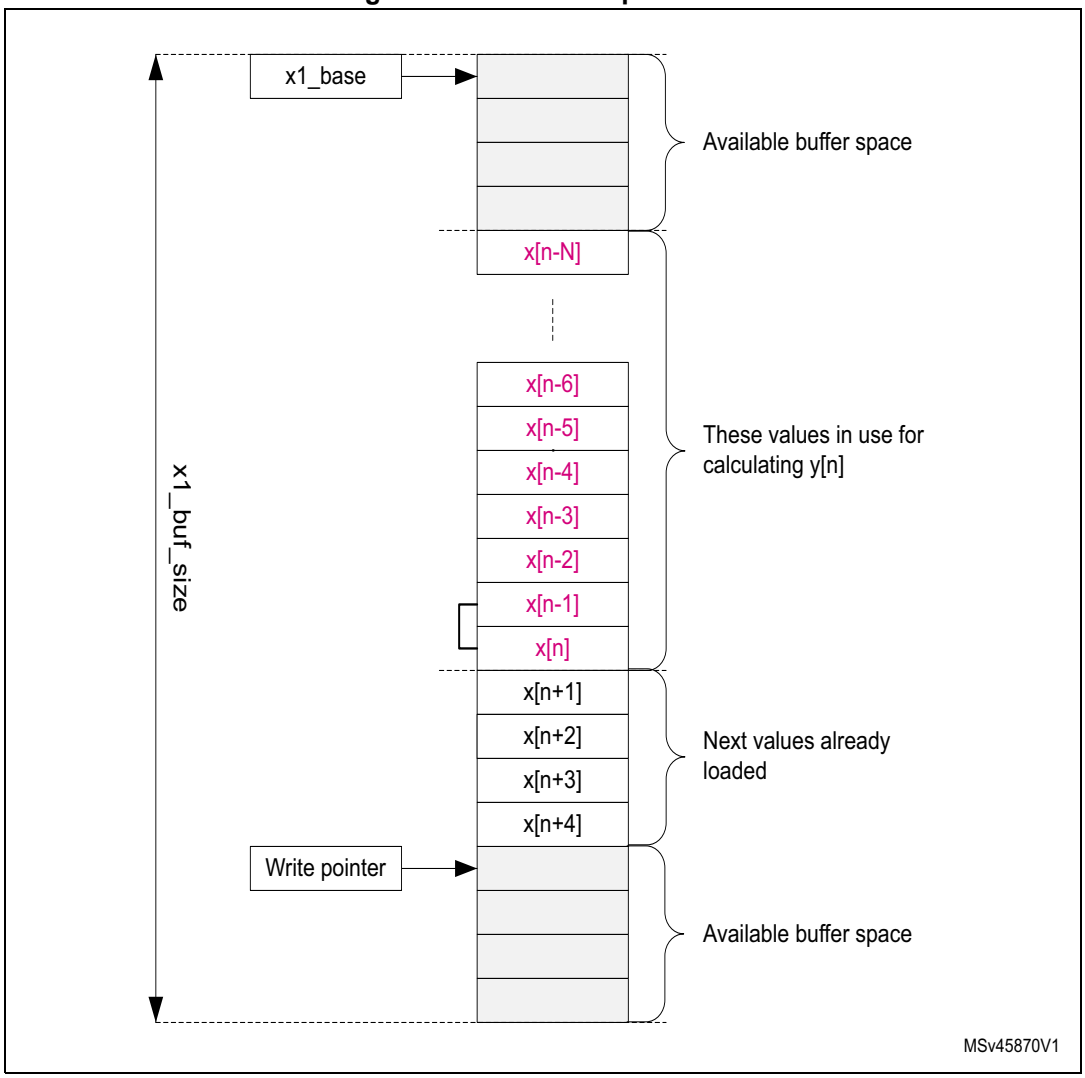
The X1 and X2 buffers are used to store data for input to the MAC. Each multiplication takes a value from the X1 buffer and a value from the X2 buffer and multiplies them together. A pointer in the control unit generates the read address offset (relative to the buffer base address) for each value. The pointers are managed by hardware according to the current function.

Figure 146. Input buffer areas



The X1 buffer can be used as a circular buffer, in which case new data are continually transferred into the input buffer whenever space is available. Pre-loading this buffer is optional for digital filters, since if no input samples have been written in the buffer when the operation is started, it is flagged as empty, which triggers the CPU or DMA to load new samples until there are enough to begin operation. Pre-loading is nevertheless useful in the case of a vector operation, that is, the input data is already available in system memory and circular operation is not required.

Figure 147. Circular input buffer



The X2 buffer can only be used in vector mode (that is not circular), and needs to be pre-loaded, except if the contents of the buffer do not change from one operation to the next. For filter functions, the X2 buffer is used to store the filter coefficients.

When operating as a circular buffer, the space allocated to the buffer (x1_buf_size) must generally be bigger than the number of elements in use for the current calculation, so that there are always new values available in the buffer. *Figure 147* illustrates the layout of the buffer for a filter operation. While calculating an output sample y[n], the unit uses a set of N+1 input samples, x[n-N] to x[n]. When this is finished, the unit starts the calculation of y[n+1], using the set of input samples x[n-N+1] to x[n+1]. The least-recent input sample, x[n-N], drops out of the input set, and a new sample, x[n+1], is added to it.

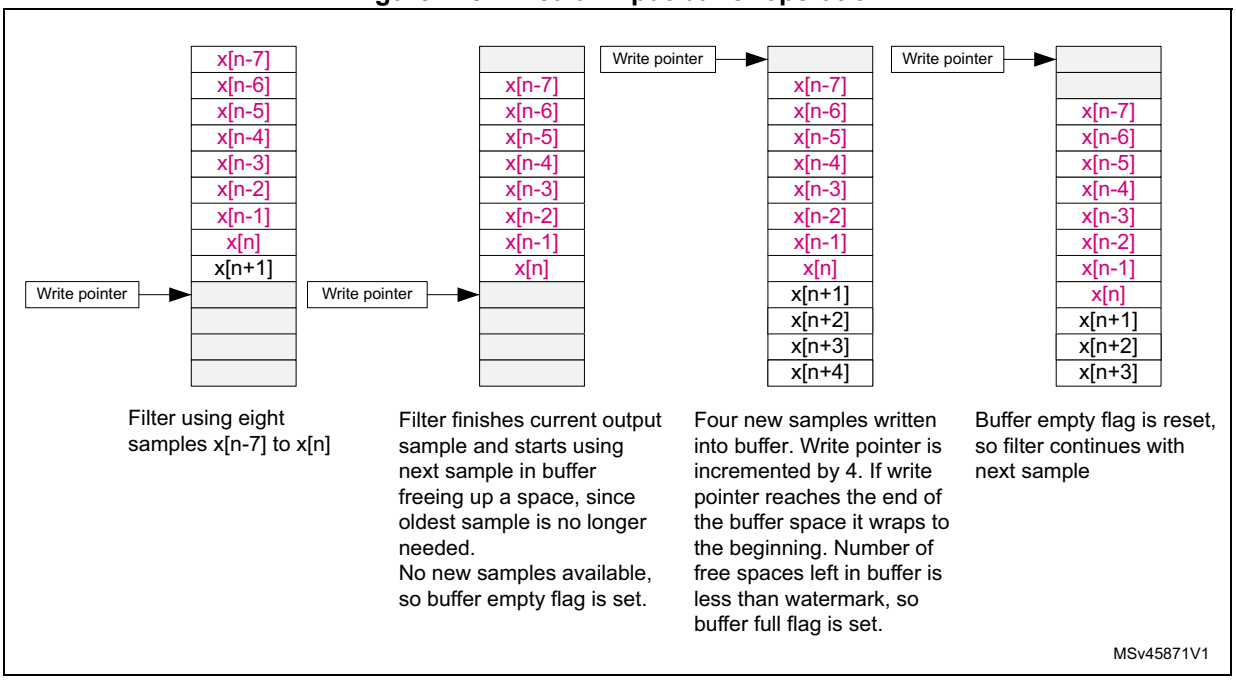
The processor, or DMA controller, must ensure that the new sample x[n+1] is available in the buffer space when required. If not, the buffer is flagged as empty, which stalls the execution of the unit until a new sample is added. No underflow condition is signaled on the X1 buffer.

Note: If the flow of samples is controlled by a timer or other peripheral such as an ADC, the buffer regularly goes empty, since the filter processes each new sample faster than the source can provide it. This is an essential feature of filter operation.

If the number of free spaces in the buffer is less than the watermark threshold programmed in the FULL_WM bitfield of the FMAC_X1BUFCFG register, the buffer is flagged as full. As long as the full flag is not set, interrupts are generated, if enabled, to request more data for the buffer. The watermark allows several data to be transferred under one interrupt, without danger of overflow. Nevertheless, if an overflow does occur, the OVFL error flag is set and the write data is ignored. The write pointer is not incremented in the event of an overflow.

The operation of the X1 buffer during a filtering operation is illustrated in *Figure 148*. This example shows an 8-tap FIR filter with a watermark set to four.

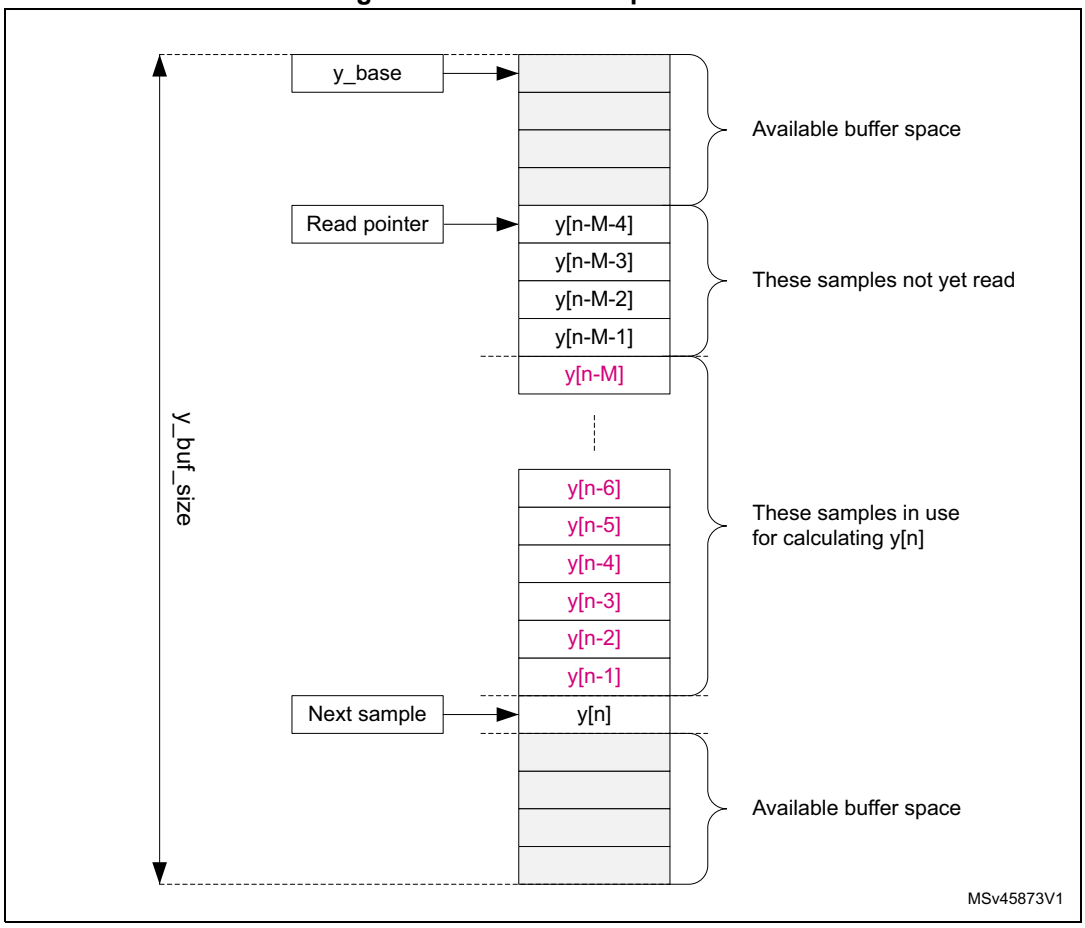
Figure 148. Circular input buffer operation



19.3.4 Output buffer

The Y (output) buffer is used to store the output of an accumulation. Each new output value is stored in the buffer until it is read by the processor or DMA controller. Each time a read access is made to the read data register, the read data is fetched from the address indicated by the read pointer. This pointer is incremented after each read, and wraps back to the base address when it reaches the end of the allocated Y buffer space.

Figure 149. Circular output buffer



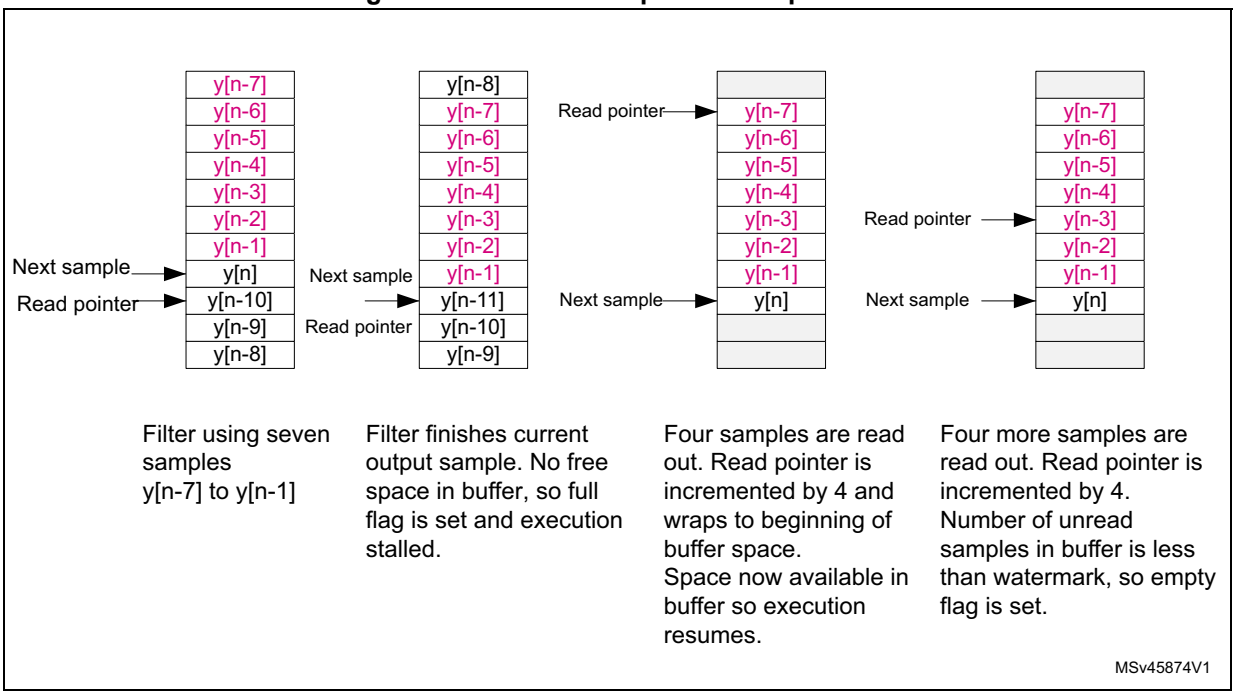
The Y buffer can also operate as a circular buffer. If the address for the next output value is the same as that indicated by the read pointer (an unread sample), then the buffer is flagged as full and execution stalled until the sample is read.

In the case of IIR filters, the Y buffer is used to store the set of M previous output samples, y[n-M] to y[n-1], used for calculating the next output sample y[n]. Each time a new sample is added to the set, the least recent sample y[n-M] drops out.

If the number of unread data in the buffer is less than the watermark threshold programmed in the EMPTY_WM bitfield of the FMAC_YBUFCFG register, the buffer is flagged as empty. As long as the empty flag is not set, interrupts or DMA requests are generated, if enabled, to request reads from the buffer. The watermark allows several data to be transferred under one interrupt, without danger of underflow. Nevertheless, if an underflow does occur, the UNFL error flag is set. In this case, the read pointer is not incremented and the read operation returns the content of the memory at the read pointer address.

The operation of the Y buffer in circular mode is illustrated in *Figure 150*. This example shows a 7-tap IIR filter with a watermark set to four.

Figure 150. Circular output buffer operation



19.3.5 Initialization functions

The following functions initialize the FMAC unit. They are triggered by writing the appropriate value in the FUNC bitfield of the FMAC_PARAM register, with the START bit set. The P and Q bitfields must also contain the appropriate parameter values for each function as detailed below. The R bitfield is not used. When the function completes, the START bit is automatically reset by hardware.

During initialization, it is recommended that the DMA requests and interrupts be disabled. The transfer of data into the FMAC memory can be done by software or by memory-to-memory DMA transfers, since no flow control is required.

Load X1 buffer

This function pre-loads the X1 buffer with N values, starting from the address in X1_BASE. Successive writes to the FMAC_WDATA register load the write data into the X1 buffer and increment the write address. The write pointer points to the address X1_BASE + N when the function completes.

The function can be used to pre-load the buffer with the elements of a vector, or to initialize the input storage elements of a filter.

Parameters

- The parameter P contains the number of values, N, to be loaded into the X1 buffer.
- The parameters Q and R are not used.

The function completes when N writes have been performed to the FMAC_WDATA register.

Load X2 buffer

This function pre-loads the X2 buffer with N + M values, starting from the address in X2_BASE. Successive writes to the FMAC_WDATA register load the write data into the X2 buffer and increment the write address.

The function can be used to pre-load the buffer with the elements of a vector, or the coefficients of a filter. In the case of an IIR, the N feed-forward and M feed-back coefficients are concatenated and loaded together into the X2 buffer. The total number of coefficients is equal to N + M. For an FIR, there are no feedback coefficients, so M = 0.

Parameters

- The parameter P contains the number of values, N, to be loaded into the X2 buffer starting from address X2_BASE.
- The parameter Q contains the number of values, M, to be loaded into the X2 buffer starting from address X2_BASE + N.
- The parameter R is not used.

The function completes when N + M writes have been performed to the FMAC_WDATA register.

Load Y buffer

This function pre-loads the Y buffer with N values, starting from the address in Y_BASE. Successive writes to the FMAC_WDATA register load the write data into the Y buffer and increment the write address. The read pointer points to the address Y_BASE + N when the function completes.

The function can be used to pre-load the feedback storage elements of an IIR filter.

Parameters

- The parameter P contains the number of values to be loaded into the Y buffer.
- The parameters Q and R are not used.

The function completes when N writes have been performed to the FMAC_WDATA register.

19.3.6 Filter functions

The following filter functions are supported by the FMAC unit. These functions are triggered by writing the corresponding value in the FUNC bitfield of the FMAC_PARAM register with the START bit set. The P, Q and R bitfields must also contain the appropriate parameter values for each function as detailed below. The filter functions continue to run until the START bit is reset by software.

Convolution (FIR filter)

Y = **B***__X__

$$y_n = 2^R \cdot \sum_{k=0}^{N} b_k x_{n-k}$$

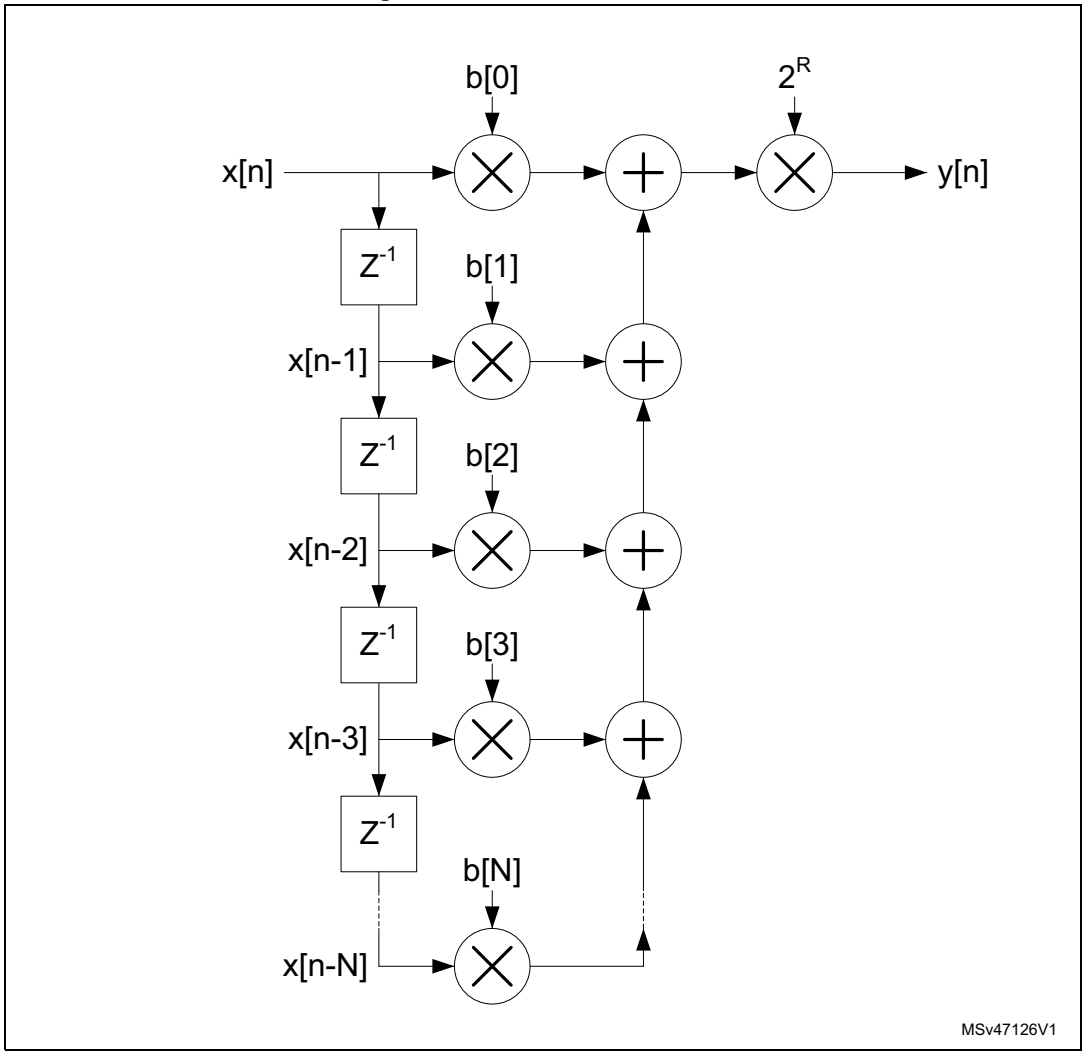
This function performs a convolution of a vector **B** of length N+1 and a vector **X** of indefinite length. The elements of **Y** for incrementing values of n are calculated as the dot product,

$y_n = \underline{\mathbf{B}}.\underline{\mathbf{X}}_{\mathbf{n}}$, where $\underline{\mathbf{X}}_{\mathbf{n}} = [x_{n-N},...,x_n]$ is composed of the N+1 elements of **X** at indexes n - N to n.

This function corresponds to a finite impulse response (FIR) filter, where vector **B** contains the filter coefficients and vector **X** the sampled data.

The structure of the filter (direct form) is shown in *Figure 151*.

Figure 151. FIR filter structure



Note that the cross correlation vector can be calculated by reversing the order of the coefficient vector **B**.

Input:

- X1 buffer contains the elements of vector **X**. It is a circular buffer of length N + 1 + d.
- X2 buffer contains the elements of vector **B**. It is a fixed buffer of length N + 1.

Output:

- Y buffer contains the output values, y_n. It is a circular buffer of length d.

Parameters:

- The parameter P contains the length, N+1, of the coefficient vector **B** in the range [2:127].
- The parameter R contains the gain to be applied to the accumulator output. The value output to the Y buffer is multiplied by 2^R, where R is in the range [0:7]
- The parameter Q is not used.

The function completes when the START bit in the FMAC_PARAM register is reset by software.

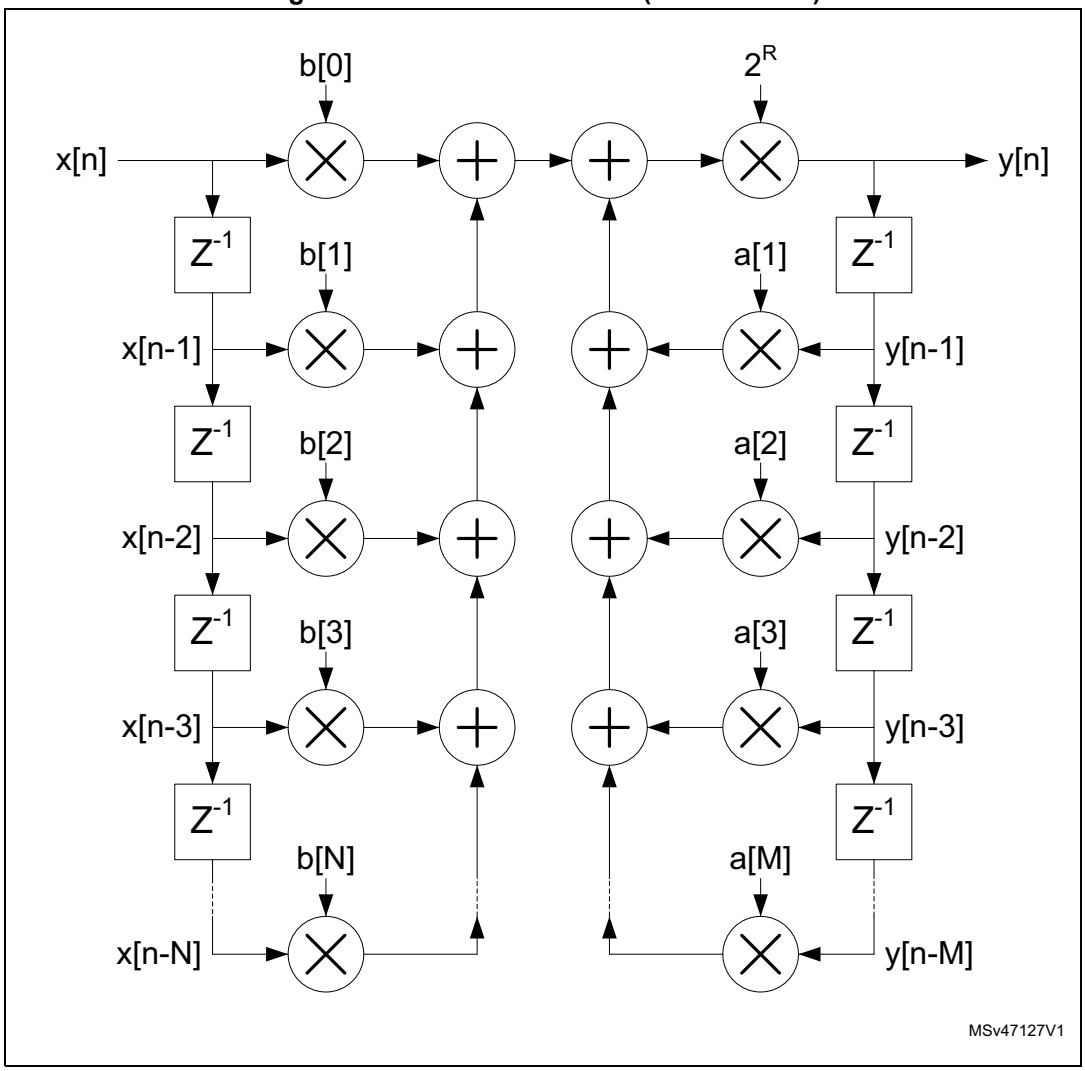
IIR filter

Y = B*X+A*Y'

$$y_n = 2^R \cdot \left(\sum_{k=0}^{N} b_k x_{n-k} + \sum_{k=1}^{M} a_k y_{n-k} \right)$$

This function implements an infinite impulse response (IIR) filter. The filter output vector **Y** is the convolution of a coefficient vector **B** of length N+1 and a vector **X** of indefinite length, plus the convolution of the delayed output vector **Y'** with a second coefficient vector **A**, of length M. The elements of **Y** for incrementing values of n are calculated as y_n = **B**.$\mathbf{X_n}$ + **A**.$\mathbf{Y_{n-1}}$, where $\mathbf{X_n}$ = $[x_{n-N},...,x_n]$ comprises the N+1 elements of **X** at indexes n - N to n, while $\mathbf{Y_{n-1}}$ = $[y_{n-M},...,y_{n-1}]$ comprises the M elements of **Y** at indexes n - M to n - 1. The structure of the filter (direct form 1) is shown in *Figure 152*.

Figure 152. IIR filter structure (direct form 1)



Input:

- X1 buffer contains the elements of vector **X**. It is a circular buffer of length N + 1+ d.
- X2 buffer contains the elements of coefficient vectors **B** and **A** concatenated (b_0, b_1, b_2..., b_N, a_1, a_2, ..., a_M). It is a fixed buffer of length M+N+1.

Output:

- Y buffer contains the output values, y_n. It is a circular buffer of length M + d.

Parameters

- The parameter P contains the length, N + 1, of the coefficient vector **B** in the range [2:64].
- The parameter Q contains the length, M, of the coefficient vector **A** in the range [1:63].
- The parameter R contains the gain to be applied to the accumulator output. The value output to the Y buffer is multiplied by 2^R, where R is in the range [0:7].

The function completes when the START bit in the FMAC_PARAM register is reset by software.

19.3.7 Fixed point representation

The FMAC operates in fixed point signed integer format. Input and output values are q1.15.

In q1.15 format, numbers are represented by one sign bit and 15 fractional bits (binary decimal places). The numeric range is therefore -1 (0x8000) to 1 - 2^{-15} (0x7FFF).

The accumulator has 26 bits, of which 22 are fractional and 4 are integer/sign (q4.22). This allows it to support partial accumulation sums in the range -8 (0x2000000) to +7.99999976 (0x1FFFFFF). A programmable gain from 0dB to 42dB in steps of 6dB can be applied at the output of the accumulator.

Note that the content of the accumulator is not saturated if the numeric range is exceeded. Partial sums whose value is greater than +7.99999976 or less than -8, wrap but this is harmless provided subsequent accumulations undo the wrapping. Nevertheless, the SAT flag in the FMAC_SR register is set if wrapping occurs, and generates an interrupt if the SATIEN bit is set in the FMAC_CR register. This helps in debugging the filter.

The data output by the accumulator can optionally be saturated, after application of the programmable gain, by setting the CLIPEN bit in the FMAC_CR register. If this bit is set, then any value which exceeds the numeric range of the q1.15 output, is set to 1 - 2^{-15} or -1, according to the sign. If clipping is not enabled, the unused accumulator bits after applying the gain is simply truncated.

19.3.8 Implementing FIR filters with the FMAC

The FMAC supports FIR filters of length N, where N is the number of taps or coefficients. The minimum local memory requirement for a FIR filter of length N is 2N + 1:

– N coefficients
– N input samples
– 1 output sample

Since the local memory size is 256, the maximum value for N is 127.

If maximum throughput is required, it may be necessary to allocate a small amount of extra space, d1 and d2, to the input and output sample buffers respectively, to ensure that the filter never stalls waiting for a new input sample, or waiting for the output sample to be read. In this case, the local memory requirement is 2N + d1 + d2.

The buffers must be configured as follows:

- X1_BUF_SIZE = N + d1;
- X2_BUF_SIZE = N;
- Y_BUF_SIZE = d2 (or 1 if no extra space is required)

The buffer base addresses can be allocated anywhere, but the X2 buffer must not overlap with the others, or else the coefficients are overwritten. An example configuration is:

- X2_BASE = 0;
- X1_BASE = N;
- Y_BASE = 2N + d1

However, if the memory space is limited, the X1 and Y buffer areas can be overlapped, such that each output sample takes the place of the oldest input sample, which is no longer required:

- X2_BASE = 0;
- X1_BASE = N;
- Y_BASE = N

In this case, Y_BUF_SIZE = X1_BUF_SIZE = N + d1, so that the buffers remain in sync.

Note: The FULL_WM bitfield of X1 buffer configuration register must be programmed with a value less than or equal to $log_2(d1)$, otherwise the buffer is flagged full before N input samples have been written, and no more samples are requested. Similarly, the EMPTY_WM bitfield of the Y buffer configuration register must be less than or equal to $log_2(d2)$.

The filter coefficients **must** be pre-loaded into the X2 buffer, using the Load X2 Buffer function. The X1 buffer can optionally be pre-loaded with any number of samples up to a maximum of N. There is no point in pre-loading the Y buffer, since for the FIR filter there is no feedback path.

After configuring and initializing the buffers, the FMAC_CR register must be programmed according to the method used for writing and reading data to and from the FMAC memory.

Three methods are supported:

- Polling: No DMA request or Interrupt request is generated. Software must check that the X1_FULL flag is low before writing to WDATA, or that the Y_EMPTY flag is low before reading from RDATA.
- Interrupt: The interrupt request is asserted while the X1_FULL flag is low, for writes, or when the Y_EMPTY flag is low, for reads.
- DMA: DMA requests are asserted on the DMA write channel while the X1_FULL flag is low, and on the read channel while the Y_EMPTY flag is low.

Different methods can be used for read and for write. However it is not recommended to use both interrupts and DMA requests for the same operation[(a)]. The valid combinations are listed in *Table 132*.

Table 132. Valid combinations for read and write methods

WIEN	RIEN	DMAWEN	DMAREN	Write	Read
0	0	0	0	Polling	Polling
0	1	0	0	Polling	Interrupt
1	0	0	0	Interrupt	Polling
1	1	0	0	Interrupt	Interrupt
0	0	0	1	Polling	DMA
0	0	1	0	DMA	Polling
0	0	1	1	DMA	DMA
0	1	1	0	DMA	Interrupt
1	0	0	1	Interrupt	DMA

a. If both interrupts and DMA requests are enabled then only DMA must perform the transfer.

The filter is started by writing to the FMAC_PARAM register with the following bitfield values:

- FUNC = 8 (FIR filter);
- P = N (number of coefficients);
- Q = “Don’t care”;
- R = Gain;
- START = 1;

If less than N + d - 2^{FULL_WM} values have been pre-loaded in the X1 buffer, the X1FULL flag remains low. If the WIEN bit is set in the FMAC_CR register, then the interrupt request is asserted immediately to request the processor to write 2^{FULL_WM} additional samples into the buffer, via the FMAC_WDATA register. It remains asserted until the X1FULL flag goes high in the FMAC_SR register. The interrupt service routine must check the X1FULL flag after every 2^{FULL_WM} writes to the FMAC_WDATA register, and repeat the transfer until the flag goes high. Similarly, if the DMAWEN bit is set in the FMAC_CR register, DMA write channel requests are generated until the X1FULL flag goes high.

The filter calculates the first output sample when at least N samples have been written into the X1 buffer (including any pre-loaded samples).

When 2^{EMPTY_WM} output samples have been written into the Y buffer, the YEMPTY flag in the FMAC_SR register goes low. If the RIEN bit is set in the FMAC_CR register, the interrupt request is asserted to request the processor to read 2^{EMPTY_WM} samples from the buffer, via the FMAC_RDATA register. It remains asserted until the YEMPTY flag goes high. The interrupt service routine must check the YEMPTY flag after every 2^{EMPTY_WM} reads from the FMAC_RDATA register, and repeat the transfer until the flag goes high. If the DMAREN bit is set in the FMAC_CR, DMA read channel requests are generated until the YEMPTY flag goes high.

The filter continues to operate in this fashion until it is stopped by the software resetting the START bit.

19.3.9 Implementing IIR filters with the FMAC

The FMAC supports IIR filters of length N, where N is the number of feed-forward taps or coefficients. The number of feedback coefficients, M, can be any value from 1 to N-1. Only direct form 1 implementations can be realized, so filters designed for other forms need to be converted.

The minimum memory requirement for an IIR filter with N feed-forward coefficients and M feed-back coefficients is 2N + 2M:

- N + M coefficients
- N input samples
- M output samples

If M = N-1, then the maximum filter length that can be implemented is N = 64.

As for the FIR, for maximum throughput, a small amount of additional space, d1 and d2, is allowed in the input and output buffer size respectively, making the total memory requirement 2M + 2N + d1 + d2.

The buffers must be configured as follows:

- X1_BUF_SIZE = N + d1;
- X2_BUF_SIZE = N + M;
- Y_BUF_SIZE = M + d2;

The buffer base addresses can be allocated anywhere, but must not overlap. An example configuration is given below:

- X2_BASE = 0;
- X1_BASE = N + M;
- Y_BASE = 2N + M + d1;

Note: The FULL_WM bitfield of X1 buffer configuration register must be programmed with a value less than or equal to $log_2(d1)$, otherwise the buffer is flagged full before N input samples have been written, and no more samples are requested. Similarly, the EMPTY_WM bitfield of the Y buffer configuration register must be less than or equal to $log_2(d2)$.

The filter coefficients (N feed-forward followed by M feedback) **must** be pre-loaded into the X2 buffer, using the Load X2 Buffer function. The X1 buffer can optionally be pre-loaded with any number of samples up to a maximum of N. The Y buffer can optionally be pre-loaded with any number of values up to a maximum of M. This has the effect of initializing the feedback delay line.

After configuring the buffers, the FMAC_CR register must be programmed in the same way as for the FIR filter (see *Section 19.3.8: Implementing FIR filters with the FMAC*).

The filter is started by writing to the FMAC_PARAM register with the following bitfield values:

- FUNC = 9 (IIR filter);
- P = N (number of feed-forward coefficients);
- Q = M (number of feed-back coefficients);
- R = Gain;
- START = 1;

If less than N + d - 2^{FULL_WM} values have been pre-loaded in the X1 buffer, the X1FULL flag remains low. If the WIEN bit is set in the FMAC_CR register, then the interrupt request is asserted immediately to request the processor to write 2^{FULL_WM} additional samples into the buffer, via the FMAC_WDATA register. It remains asserted until the X1FULL flag goes high in the FMAC_SR register. The interrupt service routine must check the X1FULL flag after every 2^{FULL_WM} writes to the FMAC_WDATA register, and repeat the transfer until the flag goes high. Similarly, if the DMAWEN bit is set in the FMAC_CR register, DMA write channel requests are generated until the X1FULL flag goes high.

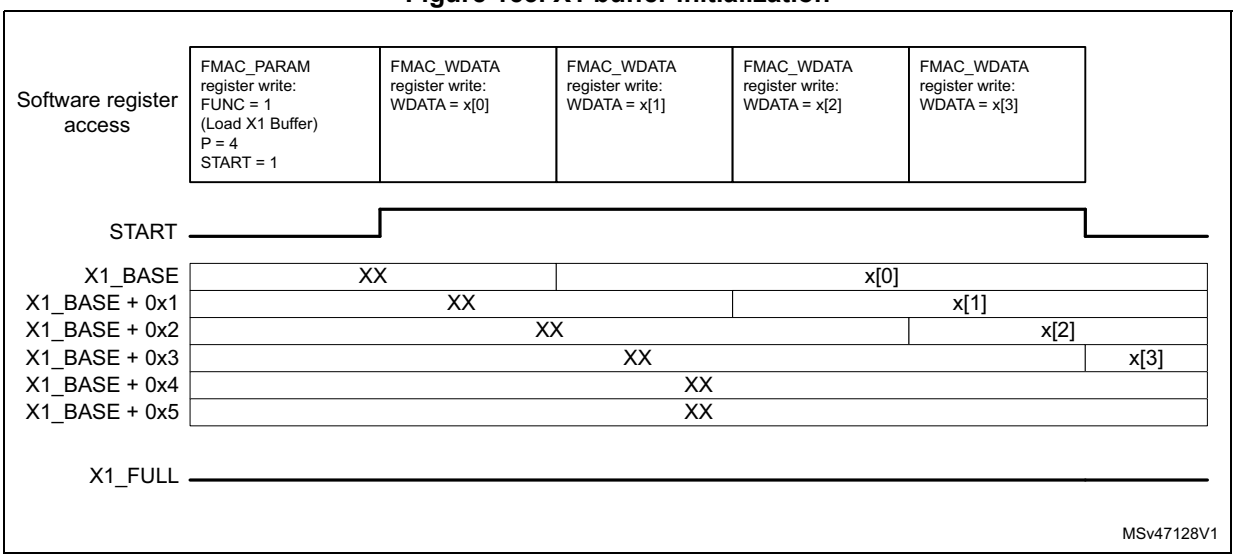
The filter calculates the first output sample when at least N samples have been written into the X1 buffer (including any pre-loaded samples). The first sample is calculated using the first N samples in the X1 buffer, and the first M samples in the Y buffer (whether or not they are preloaded. The first output sample is written into the Y buffer at Y_BASE + M.

When 2^{EMPTY_WM} new output samples have been written into the Y buffer, the YEMPTY flag in the FMAC_SR register goes low. If the RIEN bit is set in the FMAC_CR register, the interrupt request is asserted to request the processor to read 2^{EMPTY_WM} samples from the buffer, via the FMAC_RDATA register. It remains asserted until the YEMPTY flag goes high. The interrupt service routine must check the YEMPTY flag after every 2^{EMPTY_WM} reads from the FMAC_RDATA register, and repeat the transfer until the flag goes high. If the DMAREN bit is set in the FMAC_CR, DMA read channel requests are generated until the YEMPTY flag goes high

The filter continues to operate in this fashion until it is stopped by the software resetting the START bit.

19.3.10 Examples of filter initialization

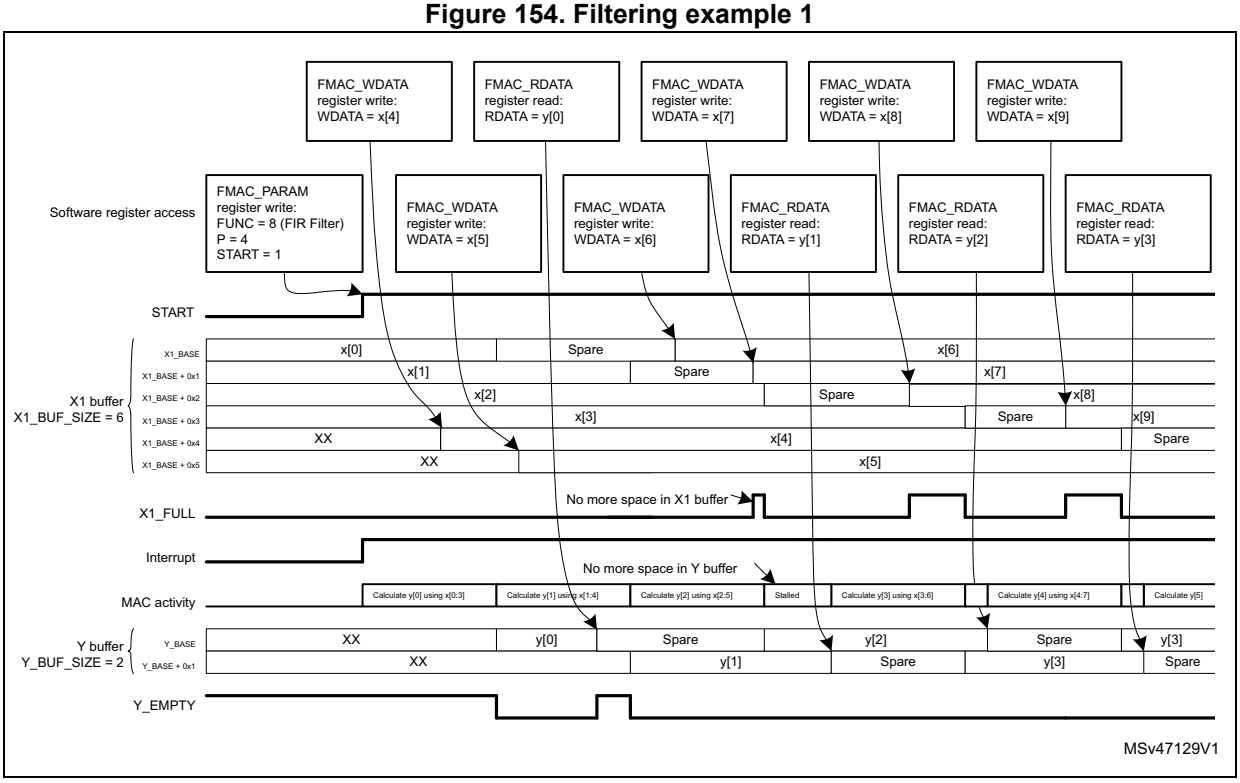
Figure 153. X1 buffer initialization



The example in *Figure 153* illustrates an X1 buffer pre-load with four samples (P = 4). The buffer size is six (X1_BUF_SIZE = 6). The initialization is launched by programming the FMAC_PARAM register with the START bit set. The four samples are then written to FMAC_WDATA, and transferred into local memory from X1_BASE onwards. The START bit resets after the fourth sample has been written. At this point, the X1 buffer contains the four samples, in order of writing, and the write pointer (next empty space) is at X1_BASE + 0x4.

19.3.11 Examples of filter operation

Figure 154. Filtering example 1



The example in *Figure 154* illustrates the beginning of a filter operation. The filter has four taps (P=4). The X1 buffer size is six and the Y buffer size is two. The FULL_WM and EMPTY_WM bitfields are both set to 0. Prior to starting the filter, the X1 buffer has been pre-loaded with four samples, x[0:3] as in *Figure 153*. So the filter starts calculating the first output sample, y[0], immediately after the START bit is set. Since the X1FULL flag is not set (due to two uninitialized spaces in the X1 buffer), the interrupt is asserted straight away, to request new data. The processor writes two new samples, x[4] and x[5], to the FMAC_WDATA register, which are transferred to the empty locations in the X1 buffer.

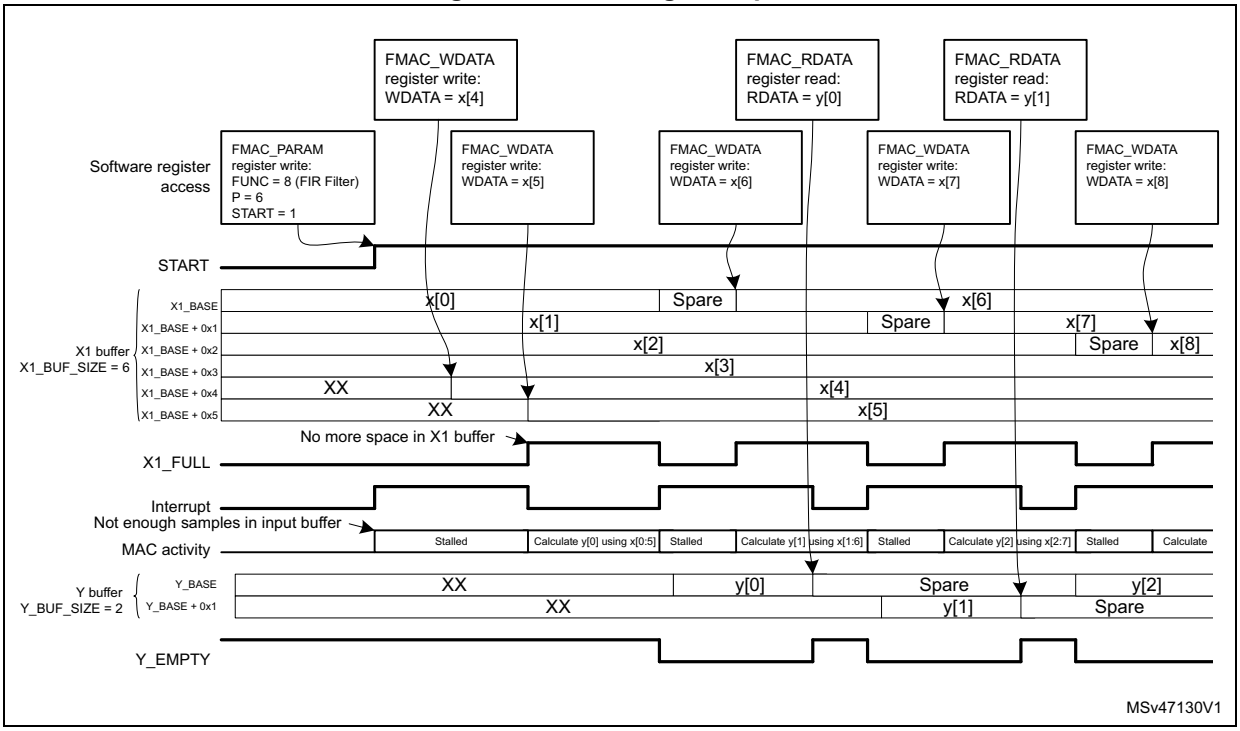
In the mean time, the FMAC finishes calculating the first output sample, y[0], and writes it into the Y buffer, causing the Y_EMPTY flag to go low. At the same time, the x[0] sample is discarded, as it is no longer required, freeing up its location in memory (at X1_BASE). The FMAC can immediately start work on the second output sample, y[1], since all the required input samples x[1:5] are present in the X1 buffer.

Since the Y_EMPTY flag is low, the interrupt remains active after the processor finishes writing x[5]. The processor reads y[0] from the FMAC_RDATA register, freeing up its location in the Y buffer. There are now no samples in the output buffer since y[1] is still being calculated, so the Y_EMPTY flag goes high. Nevertheless, the interrupt remains active, because there is still free space in the X1 buffer, which the processor next fills with x[6], and so on.

Note: In this example, the processor can fill the input buffer more quickly than the FMAC can process them, so the X1_full flag regularly goes active. However, it struggles to read the Y buffer fast enough, so the FMAC stalls regularly waiting for space to be freed up in the Y buffer. This means the filter is not executing at maximum throughput. The reason is that the

filter length is small and the processor relatively slow, in this example. So increasing the Y buffer size does not help.

Figure 155. Filtering example 2



The example in *Figure 155* illustrates the beginning of the same filter operation, but this time the filter has six taps (P=6). The X1 buffer size is six and the Y buffer size is two. The FULL_WM and EMPTY_WM bitfields are both set to 0. Prior to starting the filter, the X1 buffer has been pre-loaded with four samples, x[0:3] as in *Figure 153*. Because there are not enough samples in the input buffer, the X1FULL flag is not set, so the interrupt is asserted straight away, to request new data. The FMAC is stalled.

The processor writes two new samples, x[4] and x[5], to the FMAC_WDATA register, which are transferred to the empty locations in the X1 buffer. As soon as there are six unused samples in the X1 buffer, the X1_FULL flag goes active (since the buffer size is six), causing the interrupt to go inactive. The FMAC starts calculating the first output sample, y[0]. Since this requires all six input samples, there are no free spaces in the X1 buffer and so the X1_FULL flag remains active. Only when the FMAC finishes calculating y[0] and writes it into the Y buffer, can x[0] be discarded, freeing up a space in the X1 buffer, and deasserting X1_FULL. At the same time, the Y_EMPTY flag goes inactive. Both these flag states cause the interrupt to be asserted, requesting the processor to write a new input sample, first of all, and then read the output sample just calculated. The FMAC remains stalled until a new input sample is written.

In this example, the processor has to wait for the FMAC to finish calculating the current output sample, before it can write a new input sample, and therefore the X1 buffer regularly goes empty, stalling the FMAC. This can be avoided by allowing some extra space in the input buffer.

19.3.12 Filter design tips

The FMAC architecture imposes some constraints detailed below, on the design of digital filters.

1. Implementation of direct form 2, or transposed forms, is not efficient. Filters which have been designed for such forms must be converted to direct form 1.
2. Cascaded filters must either be combined into a single stage, or implemented as separate filters. In the latter case, multiple sets of filter coefficients can be pre-loaded into the memory, one set per stage, and only the X2_BASE address changed to select which set is used. The most efficient method of implementing a multi-stage filter is to pre-load a large X1 buffer with input samples, run the IIR filter function on it using the first stage coefficients, and store the output samples back in memory. Then change the X2_BASE pointer to point to the 2nd stage coefficients, and reload the input buffer with the output of the first stage (with a gain if required), before running the IIR function again. The procedure is repeated for all stages. Once the final stage samples have been transferred back into system memory, the input buffer can be loaded with the next set of input samples, and a new round of calculations started. Note that the N sample input buffer of each stage must be pre-loaded first of all with the N-1 last inputs from the previous round, plus one new sample, in order to keep continuity between each round. Similarly, the output buffer of each stage must be loaded with the last M samples from the previous round, for the same reason.
3. The use of direct form 1 for IIR designs can lead to large positive or negative partial sums in the accumulator, if for example a large step occurs on the input, or some of the filter coefficients' absolute values are >1. Since the accumulator is limited to 26 bits, the biggest value that it can handle without wrapping (changing sign) is 0x1FFFFFF positive or 0x2000000 negative. This corresponds to 3.99999988 and -4 respectively in q3.23 fixed point format. Wrapping does not represent a problem provided the wrapping is "undone" before the end of the accumulation. However this is not always the case when a filter is starting up and can lead to unexpected results. Consider pre-loading the output buffer with suitable values to avoid this.
4. The IIR filter has feed-forward (numerator) coefficients $[b_0, b_1, ..., b_{N-1}]$, and feed-back (denominator) coefficients $[1, a_1, ..., a_M]$. Many IIR filters require some of the denominator coefficients to have an absolute value greater than 1 to achieve a steep roll-off in the frequency response. Given that the coefficients are coded in fixed point q1.15 format, this is not possible. Nevertheless, by scaling the denominator coefficients by a factor 2^{-R}, such that $2^{-R}.[1, a_1, ..., a_M]$ are all less than 1, such filters can be implemented. However an inverse gain of 2^R must be applied at the output of the accumulator to compensate the scaling. This has an adverse effect on the signal-to-noise ratio.

19.4 FMAC registers

19.4.1 FMAC X1 buffer configuration register (FMAC_X1BUFCFG)

Address offset: 0x00

Reset value: 0x0000 0000

Access: word access

This register can only be modified if START = 0 in the FMAC_PARAM register.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X1_BUF_SIZE[7:0]								X1_BASE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 25:24 **FULL_WM[1:0]**: Watermark for buffer full flag

Defines the threshold for setting the X1 buffer full flag when operating in circular mode. The flag is set if the number of free spaces in the buffer is less than 2^{FULL_WM}.

0: Threshold = 1

1: Threshold = 2

2: Threshold = 4

3: Threshold = 8

Setting a threshold greater than 1 allows several data to be transferred into the buffer under one interrupt.

Threshold must be set to 1 if DMA write requests are enabled (DMAWEN = 1 in FMAC_CR register).

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:8 **X1_BUF_SIZE[7:0]**: Allocated size of X1 buffer in 16-bit words

The minimum buffer size is the number of feed-forward taps in the filter (+ the watermark threshold - 1).

Bits 7:0 **X1_BASE[7:0]**: Base address of X1 buffer

19.4.2 FMAC X2 buffer configuration register (FMAC_X2BUFCFG)

Address offset: 0x04

Reset value: 0x0000 0000

Access: word access

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X2_BUF_SIZE[7:0]								X2_BASE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **X2_BUF_SIZE[7:0]**: Size of X2 buffer in 16-bit words
This bitfield can not be modified when a function is ongoing (START = 1).

Bits 7:0 **X2_BASE[7:0]**: Base address of X2 buffer
The X2 buffer base address can be modified while START=1, for example to change coefficient values. The filter must be stalled when doing this, since changing the coefficients while a calculation is ongoing affects the result.

19.4.3 FMAC Y buffer configuration register (FMAC_YBUFCFG)

Address offset: 0x08

Reset value: 0x0000 0000

Access: word access

This register can only be modified if START = 0 in the FMAC_PARAM register.

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

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

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

Bits 25:24 **EMPTY_WM[1:0]**: Watermark for buffer empty flag
Defines the threshold for setting the Y buffer empty flag when operating in circular mode. The flag is set if the number of unread values in the buffer is less than 2^{EMPTY_WM}.
0: Threshold = 1
1: Threshold = 2
2: Threshold = 4
3: Threshold = 8
Setting a threshold greater than 1 allows several data to be transferred from the buffer under one interrupt.
Threshold must be set to 1 if DMA read requests are enabled (DMAREN = 1 in FMAC_CR register).

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:8 **Y_BUF_SIZE[7:0]**: Size of Y buffer in 16-bit words
For FIR filters, the minimum buffer size is 1 (+ the watermark threshold). For IIR filters the minimum buffer size is the number of feedback taps (+ the watermark threshold).

Bits 7:0 **Y_BASE[7:0]**: Base address of Y buffer

19.4.4 FMAC parameter register (FMAC_PARAM)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: word access

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

Bit 31 **START**: Enable execution
0: Stop execution
1: Start execution
Setting this bit triggers the execution of the function selected in the FUNC bitfield. Resetting it by software stops any ongoing function. For initialization functions, this bit is reset by hardware.

Bits 30:24 **FUNC[6:0]**: Function
0: Reserved
1: Load X1 buffer
2: Load X2 buffer
3: Load Y buffer
4 to 7: Reserved
8: Convolution (FIR filter)
9: IIR filter (direct form 1)
10 to 127: Reserved
This bitfield can not be modified when a function is ongoing (START = 1)

Bits 23:16 **R[7:0]**: Input parameter R.
The value of this parameter is dependent on the function.
This bitfield can not be modified when a function is ongoing (START = 1)

Bits 15:8 **Q[7:0]**: Input parameter Q.
The value of this parameter is dependent on the function.
This bitfield can not be modified when a function is ongoing (START = 1)

Bits 7:0 **P[7:0]**: Input parameter P.
The value of this parameter is dependent on the function
This bitfield can not be modified when a function is ongoing (START = 1)

19.4.5 FMAC control register (FMAC_CR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESET
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLIP EN	Res.	Res.	Res.	Res.	Res.	DMA WEN	DMA REN	Res.	Res.	Res.	SAT IEN	UNFL IEN	OVFL IEN	WIEN	RIEN
rw						rw	rw				rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RESET**: Reset FMAC unit

This resets the write and read pointers, the internal control logic, the FMAC_SR register and the FMAC_PARAM register, including the START bit if active. Other register settings are not affected. This bit is reset by hardware.

0: Reset inactive

1: Reset active

Bit 15 **CLIPEN**: Enable clipping

0: Clipping disabled. Values at the output of the accumulator which exceed the q1.15 range, wrap.

1: Clipping enabled. Values at the output of the accumulator which exceed the q1.15 range are saturated to the maximum positive or negative value (+1 or -1) according to the sign.

Bits 14:10 Reserved, must be kept at reset value.

Bit 9 **DMAWEN**: Enable DMA write channel requests

0: Disable. No DMA requests are generated

1: Enable. DMA requests are generated while the X1 buffer is not full.

This bit can only be modified when START= 0 in the FMAC_PARAM register. A read returns the current state of the bit.

Bit 8 **DMAREN**: Enable DMA read channel requests

0: Disable. No DMA requests are generated

1: Enable. DMA requests are generated while the Y buffer is not empty.

This bit can only be modified when START= 0 in the FMAC_PARAM register. A read returns the current state of the bit.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SATIEN**: Enable saturation error interrupts

0: Disabled. No interrupts are generated upon saturation detection.

1: Enabled. An interrupt request is generated if the SAT flag is set

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 3 **UNFLIEN**: Enable underflow error interrupts

0: Disabled. No interrupts are generated upon underflow detection.

1: Enabled. An interrupt request is generated if the UNFL flag is set

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 2 **OVFLIEN**: Enable overflow error interrupts
0: Disabled. No interrupts are generated upon overflow detection.
1: Enabled. An interrupt request is generated if the OVFL flag is set
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 1 **WIEN**: Enable write interrupt
0: Disabled. No write interrupt requests are generated.
1: Enabled. An interrupt request is generated while the X1 buffer FULL flag is not set.
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 0 **RIEN**: Enable read interrupt
0: Disabled. No read interrupt requests are generated.
1: Enabled. An interrupt request is generated while the Y buffer EMPTY flag is not set.
This bit is set and cleared by software. A read returns the current state of the bit.

19.4.6 FMAC status register (FMAC_SR)

Address offset: 0x14

Reset value: 0x0000 0001

Access: word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SAT	UNFL	OVFL	Res.	Res.	Res.	Res.	Res.	Res.	X1 FULL	Y EMPTY
					r	r	r							r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **SAT**: Saturation error flag
Saturation occurs when the result of an accumulation exceeds the numeric range of the accumulator.
0: No saturation detected
1: Saturation detected. If the SATIEN bit is set, an interrupt is generated.
This flag is cleared by a reset of the unit.

Bit 9 **UNFL**: Underflow error flag
An underflow occurs when a read is made from FMAC_RDATA when no valid data is available in the Y buffer.
0: No underflow detected
1: Underflow detected. If the UNFLIEN bit is set, an interrupt is generated.
This flag is cleared by a reset of the unit.

Bit 8 **OVFL**: Overflow error flag
An overflow occurs when a write is made to FMAC_WDATA when no free space is available in the X1 buffer.
0: No overflow detected
1: Overflow detected. If the OVFLIEN bit is set, an interrupt is generated.
This flag is cleared by a reset of the unit.

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

Bit 1 **X1FULL**: X1 buffer full flag

The buffer is flagged as full if the number of available spaces is less than the FULL_WM threshold. The number of available spaces is the difference between the write pointer and the least recent sample currently in use.

0: X1 buffer not full. If the WIEN bit is set, the interrupt request is asserted until the flag is set. If DMAWEN is set, DMA write channel requests are generated until the flag is set.

1: X1 buffer full.

This flag is set and cleared by hardware, or by a reset.

Note: after the last available space in the X1 buffer is filled there is a delay of 3 clock cycles before the X1FULL flag goes high. To avoid any risk of overflow it is recommended to insert a software delay after writing to the X1 buffer before reading the FMAC_SR. Alternatively, a FULL_WM threshold of 2 can be used.

Bit 0 **YEMPTY**: Y buffer empty flag

The buffer is flagged as empty if the number of unread data is less than the EMPTY_WM threshold. The number of unread data is the difference between the read pointer and the current output destination address.

0: Y buffer not empty. If the RIEN bit is set, the interrupt request is asserted until the flag is set. If DMAREN is set, DMA read channel requests are generated until the flag is set.

1: Y buffer empty.

This flag is set and cleared by hardware, or by a reset.

Note: after the last sample is read from the Y buffer there is a delay of 3 clock cycles before the YEMPTY flag goes high. To avoid any risk of underflow it is recommended to insert a software delay after reading from the Y buffer before reading the FMAC_SR. Alternatively, an EMPTY_WM threshold of 2 can be used.

19.4.7 FMAC write data register (FMAC_WDATA)

Address offset: 0x18

Reset value: 0x0000 0000

Access: word access

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **WDATA[15:0]**: Write data

When a write access to this register occurs, the write data are transferred to the address offset indicated by the write pointer. The pointer address is automatically incremented after each write access.

19.4.8 FMAC read data register (FMAC_RDATA)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: word access

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RDATA[15:0]**: Read data

When a read access to this register occurs, the read data are the contents of the Y output buffer at the address offset indicated by the READ pointer. The pointer address is automatically incremented after each read access.

19.4.9 FMAC register map

Table 133.FMAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FMAC_X1BUFCFG	Res.	Res.	Res.	Res.	Res.	Res.	FULL_WM[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	X1_BUF_SIZE[7:0]								X1_BASE[7:0]							
	Reset value							0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	FMAC_X2BUFCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	X2_BUF_SIZE[7:0]								X2_BASE[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	FMAC_YBUFCFG	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY_WM[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Y_BUF_SIZE[7:0]								Y_BASE[7:0]							
	Reset value							0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	FMAC_PARAM	START	FUNC[6:0]							R[7:0]								Q[7:0]								P[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	FMAC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESET	CLIPEN	Res.	Res.	Res.	Res.	Res.	DMAWEN	DMAREN	Res.	Res.	Res.	SATIEN	UNFLIEN	OVFLIEN	WIEN	RIEN
	Reset value																0	0						0	0				0	0	0	0	0
0x14	FMAC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAT	UNFL	OVFL	Res.	Res.	Res.	Res.	Res.	Res.	X1FULL	YEMPTY
	Reset value																						0	0	0							0	1
0x18	FMAC_WDATA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	FMAC_RDATA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

20 Hardware signal processor (HSP)

20.1 HSP introduction

The hardware signal processor (HSP) is a dedicated signal processor engine. It can offload and speed up operations initially handled by the CPU. It targets the following application areas:

- Control loops
- Metering
- Small convolutional neural networks
- General-purpose signal processing

The HSP is built around a dedicated high-performance signal processor engine (SPE) working in 32-bit floating point or 8-bit integer. The SPE runs the HSP microcode (also named firmware).

A part of the microcode (firmware) is embedded in a private ROM. This microcode provides optimized signal processing functions, named “processing functions”.

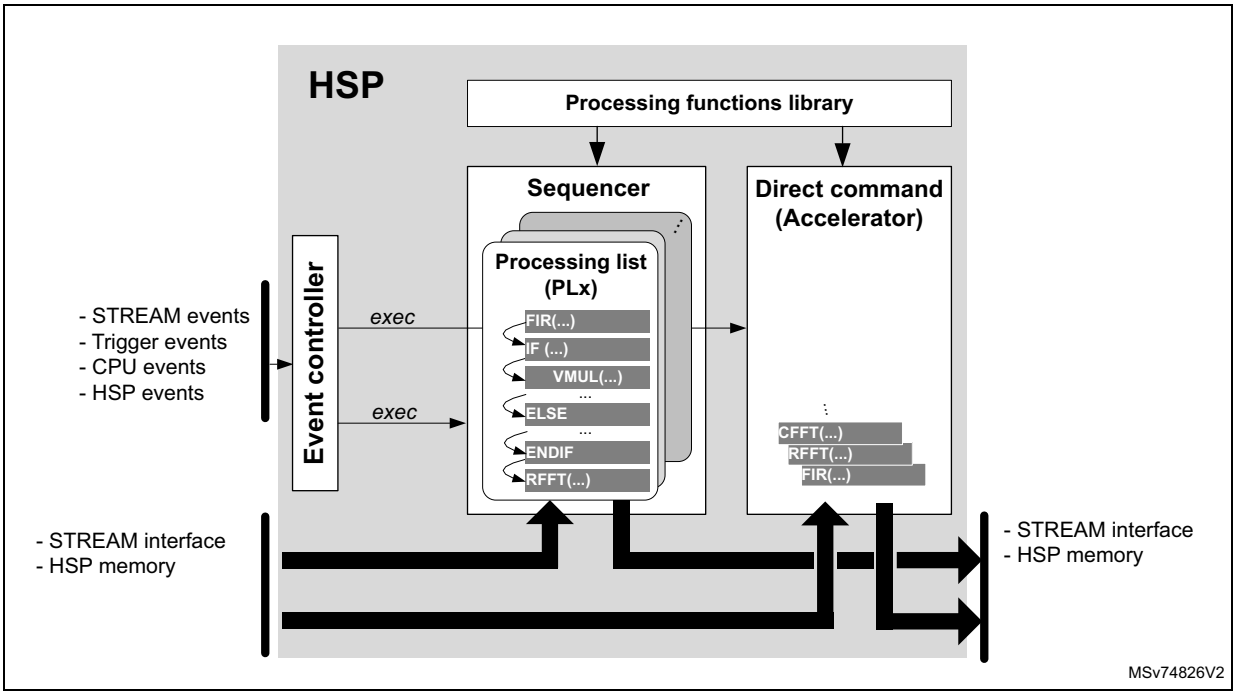
The HSP can perform sequences of operations autonomously. These sequences are called “processing lists”.

As shown in the figure below, the HSP proposes a sequencer to handle the processing lists, and a direct command interface for the accelerator function.

The peripherals connected to the event controller generate events according to their activity. These events are handled by the event controller, and the event having the highest priority activates a task. This task executes the recorded processing list. If other events are pending, the sequence is repeated when the execution of the current processing list is completed.

Each task can host a single processing list.

Figure 156. HSP functional view



The application can record up to 37 processing lists into the HSP during the runtime. A processing list represents an HSP task. The execution of processing lists is triggered by events. The event sources can be:

- STREAM interface
- Trigger signals
- Software triggers

The HSP offers the following bus interfaces:

- An AHB slave port interface giving access to the HSP control registers (SP0)
- An AHB slave port interface giving access to the HSP memories (SP1)

When the data to be processed are 16- or 32-bit integers, the HSP converts them into a 32-bit floating-point format before performing the required processing.

If the data are in 8-bit integer or 32-bit floating-point format, the HSP processes them directly.

Data exchange between the HSP and the application can be done in several ways:

- Using the BRAM as a shared memory
- Using the STREAM interface

After each processing task performed in 32-bit floating-point format, the HSP can provide the data either in floating-point or integer format. Processing performed using an 8-bit integer returns results in 8-bit integer format.

See the user manual *HSP processing functions on STM32 MCUs* (UM3360).

Finally, the HSP offers an interrupt interface allowing the application to receive interrupts from a wide variety of events.

20.2 HSP main features

- AHB slave interface for memories (SP1)
- AHB slave interface for register control (SP0)
- Rich set of signal processing commands (such as FFT, FIR, or IIR)
- Support of 2D convolution for neural networks
- Operations performed in floating-point single precision compliant with the IEEE-754 standard

  The HSP FPU is compliant with the IEEE-754 standard with the following limitations:
  - Denormals are flushed to zero.
  - Only round to the nearest is supported.
- Supported data formats:
  - Signed/unsigned 16-bit integer format
  - Signed/unsigned 32-bit integer format
  - 32-bit floating-point format
  - 8-bit integer (CNN functions)
- Trigger inputs
- Break generation
- STREAM interface
- Trigger outputs
- Interrupt services

Targeted applications:

- Audio: speech capture
- Motor control
- Digital power
- Metering

20.3 HSP implementation

The table below details the set of features implemented into the product.

Table 134. HSP features for STM32H7Power(1)

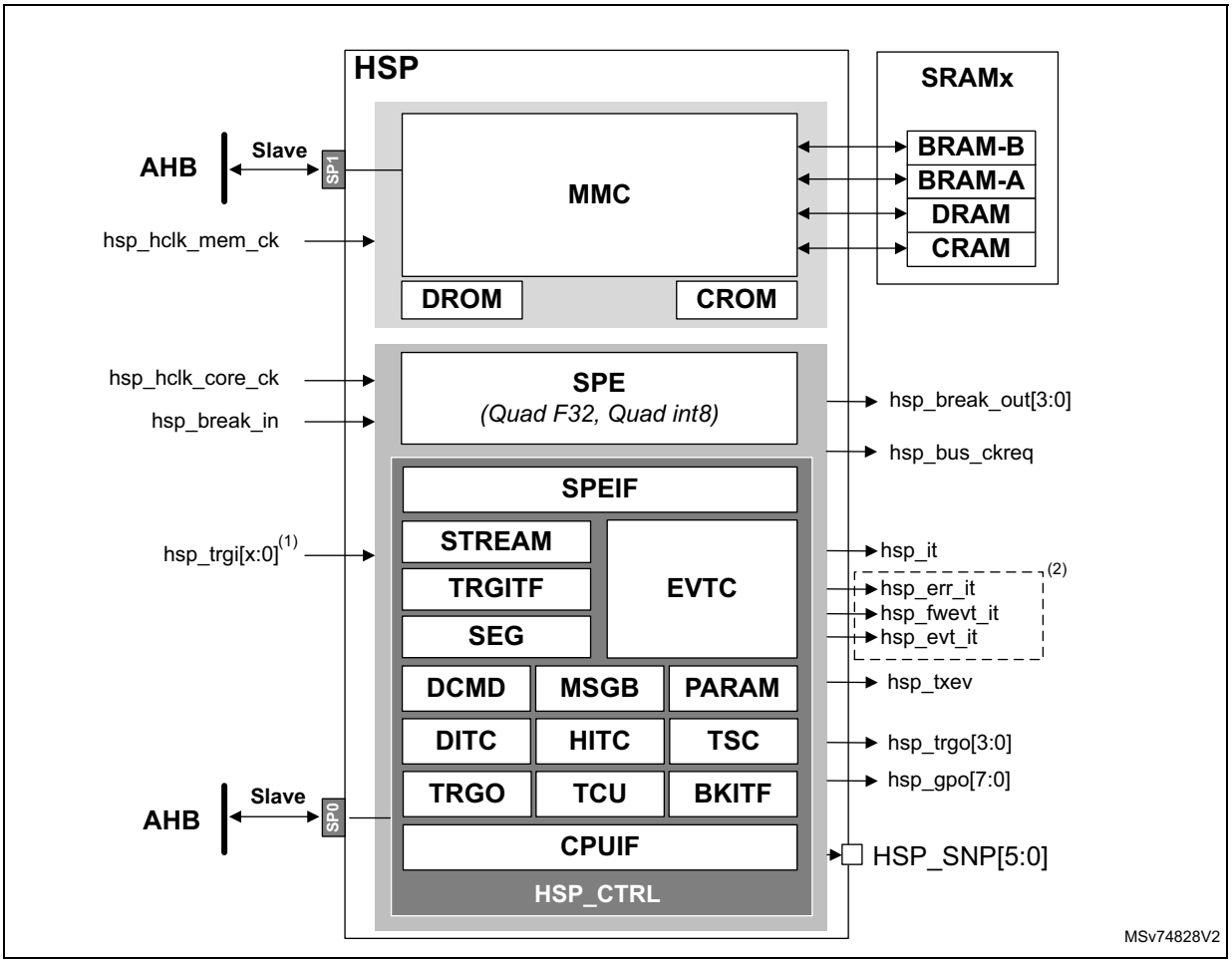
Item	Values
SRAM instance containing the BRAM, CRAM and DRAM.	AHBSRAM3
CRAM size	20 Kbytes
DRAM size	12 Kbytes
BRAM size	32 Kbytes
Global interrupt (hsp_global_it)	X
Error interrupt (hsp_err_it)	-
Event interrupt (hsp_evt_it)	-
Firmware event interrupt (hsp_fwevt_it)	-
HSPDMA and STI function	-
Number of DMA channels	-
Number of DMA requests which can be connected to peripherals	-
Number of STIAP ports	-
Number of trigger inputs	20
Direct ADC interface function	-
Number of ADC interfaces	-
Maximum number of processing lists	37

1. ‘X’ = supported, ‘-’ = not supported.

20.4 HSP functional description

20.4.1 HSP block diagram

Figure 157. HSP block diagram



1. The number of trigger inputs depends on the product, see *Section 20.3: HSP implementation*.
2. These interrupt lines are not always externally connected, see *Section 20.3: HSP implementation*. However, all interrupt source signals are internally connected to the global interrupt signal (hsp_it), which is connected to the CPU interrupt controller on all products. Please see *Section 20.4.12: HSP interrupt controller (HITC)* for details.

20.4.2 HSP pins and internal signals

Table 135. HSP external pins

Pin name	Pin type	Comment
HSP_SNP[5:0]	Output	HSP outputs for debug and diagnostics

Table 136. HSP internal signals

Signal name	Signal type	Comment
hsp_hclk_mem_ck	Input	Bus clock input for memories
hsp_hclk_core_ck	Input	Bus clock input for HSP core
hsp_bus_ckreq	Output	Bus interface clock request output
hsp_break_out[3:0]	Output	HSP break output signals
hsp_break_in	Input	HSP break input signal
hsp_it	Output	HSP global interrupt
hsp_err_it(1)	Output	HSP error interrupt
hsp_evt_it(1)	Output	HSP event interrupt
hsp_fwevt_it(1)	Output	HSP firmware event interrupt
hsp_txev	Output	HSP direct command complete event.
hsp_trgi[x:0](2)	Input	HSP trigger input signals
hsp_trgo[3:0]	Output	HSP trigger output signals
hsp_gpo[7:0]	Output	HSP general-purpose output signals
hsp_dbgmcu_stp	Input	Halt request from DBGMCU (check DBG section to see if the feature is available)

1. This interrupt line is not always connected to the interrupt controller, see *Section 20.3: HSP implementation*.
2. The number of trigger input signals depends on the product, see *Section 20.3: HSP implementation*.

The table below shows how HSP trigger inputs are connected in the product.

Table 137.HSP trigger input connections for STM32H7Power

Input trigger names	Trigger sources
hsp_trgi0	GPDMA1 dma_chan_tc[15]
hsp_trgi1	GPDMA1 dma_chan_tc[14]
hsp_trgi2	GPDMA1 dma_chan_tc[13]
hsp_trgi3	GPDMA1 dma_chan_tc[12]
hsp_trgi4	EXTI0
hsp_trgi5	NPU host_int_3
hsp_trgi6	TIM1 TRGO
hsp_trgi7	TIM1 TRGO2
hsp_trgi8	TIM2 TRGO
hsp_trgi9	TIM3 TRGO
hsp_trgi10	TIM6_TRGO
hsp_trgi11	TIM7_TRGO
hsp_trgi12	TIM8_TRGO

Table 137.HSP trigger input connections for STM32H7Power (continued)

Input trigger names	Trigger sources
hsp_trgi13	TIM8_TRGO2
hsp_trgi14	TIM15_TRGO
hsp_trgi15	HPDMA1 dma_chan_tc[15]
hsp_trgi16	HPDMA1 dma_chan_tc[14]
hsp_trgi17	HPDMA1 dma_chan_tc[13]
hsp_trgi18	HPDMA1 dma_chan_tc[12]
hsp_trgi19	ADF1_SAD_DET

The table below shows how HSP trigger outputs are connected in the product.

Table 138.HSP trigger output connections for STM32H7Power

Output trigger names	Trigger destination
hsp_trgo0	GPDMA1 dma_trig[15] HPDMA1 dma_trig[15]
hsp_trgo1	GPDMA1 dma_trig[16] HPDMA1 dma_trig[16]
hsp_trgo2	GPDMA1 dma_trig[17] HPDMA1 dma_trig[17]
hsp_trgo3	GPDMA1 dma_trig[18] HPDMA1 dma_trig[18]
hsp_gpo0	NPU ext_trigger_async[0]
hsp_gpo1	-
hsp_gpo2	-
hsp_gpo3	-
hsp_gpo4	-
hsp_gpo5	-
hsp_gpo6	-
hsp_gpo7	-

The table below shows how HSP break outputs are connected in the product.

Table 139.HSP break connections on STM32H7Power

Signal name	Break destination
hsp_break_out0	TIM1_brk1[13] TIM1_brk2[13] TIM8_brk1[13] TIM8_brk2[13] TIM20_brk1[13] TIM20_brk2[13] HRTIM1_sys_flt
hsp_break_out1	TIM1_brk1[14] TIM1_brk2[14] TIM8_brk1[14] TIM8_brk2[14] TIM20_brk1[14] TIM20_brk2[14]
hsp_break_out2	TIM15_brk1[13] TIM16_brk1[13] TIM17_brk1[13] HRTIM2_sys_flt
hsp_break_out3	TIM15_brk1[14] TIM16_brk1[14] TIM17_brk1[14]
hsp_break_in	-

20.4.3 Floating-point format

The HSP is a powerful and flexible signal processing engine, able to perform operations on data located into its shared memory (BRAM).

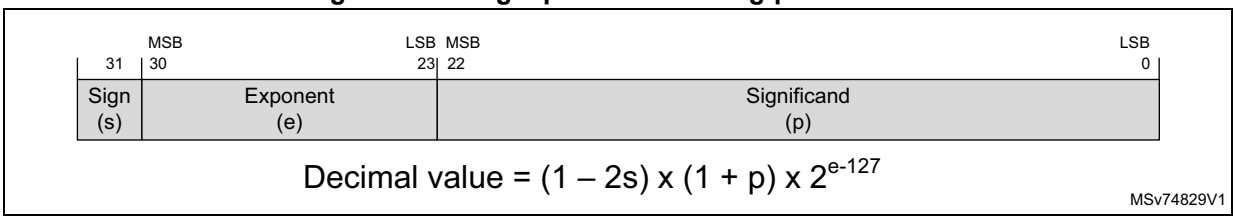
The HSP can work by processing data blocks (vector) or sample per sample. The latter mode is particularly helpful for closed-loop control.

The HSP supports several kinds of data input formats: 8-, 16-, 32-bit integers, and 32-bit floating points. For simplification, this document refers to 32-bit floating points as “float32”. In the same way, signed 8-bit integers are named “int8”.

The SPE embedded into the HSP performs its computation in single-precision floating points (float32) or 8-bit integers (int8).

As defined by the IEEE-754 standard, the single-precision floating-point numbers are stored into the memory as shown in the figure below.

Figure 158. Single-precision floating-point format



The single precision floating-point number includes:

- A sign bit
- A significand part of 23 bits
- An exponent of 8 bits (-126 to + 127)

The first leading 1 bit of a normalized significand is “hidden”, or “implicit”, extending the significand to 24-bit precision.

Floating point greatly simplifies working with large numbers, up to about $\pm 3.4e^{38}$, and small numbers down to about $\pm 1.17e^{-38}$.

The HSP embeds the capability of converting integers into float32 and vice versa.

The HSP also supports integer 8-bit operations. They are dedicated to run convolutional neural networks. In this mode, the internal accumulator size is 32 bits to keep the best accuracy.

20.4.4 Memory management

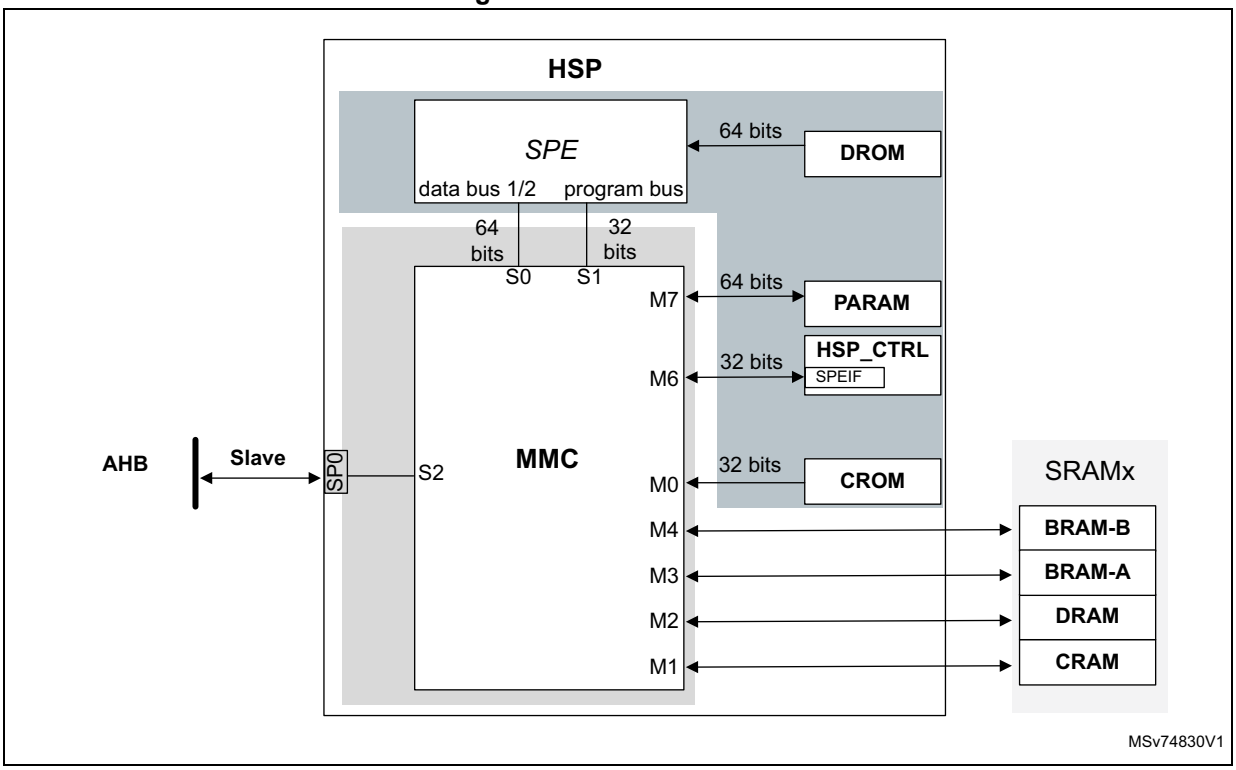
A memory controller (MMC) manages the SRAM shared with the HSP. This SRAM can be used by the application like any other SRAM or dedicated to HSP activity when the HSP is activated (see *Section 20.3: HSP implementation* to check which SRAM is shared with the HSP on the product). This SRAM is split into five different sections when the application intends to use the HSP. The different sections are listed below:

- The code RAM (CRAM) section, containing the HSP firmware program, plug-ins, and processing lists
- The data RAM (DRAM) section, containing SPE sensitive data and processing list parameters
- The buffer RAM A (BRAM-A), RAM B (BRAM-B), and RAM-AB sections, containing the working data, filter coefficients, and filter states

In addition, the HSP embeds a code ROM and a data ROM:

- The code ROM (CROM) contains the generic signal processing functions library.
- The data ROM (DROM) contains constants, such as FFT coefficients.

Figure 159. MMC overview



Memory protection and sharing

The access to the memory shared with the HSP is controlled by the MMC (memory controller) block. After a system reset, this SRAM is available for the application, and can be used as a general-purpose SRAM.

Note: This SRAM can be enabled or disabled via an RCC control bit. Make sure it is enabled before using it.

The BRAM-A, BRAM-B, and BRAM-AB sections are always shared with the application and are generally used for data exchanges and working buffers.

The application can restrict the access to CRAM and DRAM sections with the CDRLCK[1:0] bitfield in the *HSP control register (HSP_CR)*. CRAM and DRAM access filtering works as follows:

- CDRLCK[1:0] = 0: read and write access through the AHB slave of CRAM and DRAM is allowed.
- CDRLCK[1:0] = 1: write access to CRAM and DRAM is not allowed. Any read access is allowed. Only an HSP reset can restore write access to these memories.
- CDRLCK[1:0] = 2 or 3: read and write access to CRAM and DRAM is not allowed. Only an HSP reset can restore read and write access to these memories.

An invalid application read access to a RAM protected by CDRLCK[1:0] returns 0. An invalid application write access to a RAM protected by CDRLCK[1:0] is ignored. In both cases, the ACCERRF and SAHBERR flags are set.

Accessing above BRAMAB_END (see *Figure 160*) also generates a bus error. The ACCERRF and SAHBERR flags are set as well.

See *Section 20.4.12: HSP interrupt controller (HITC)*.

The access restrictions programmed with CDRLCK[1:0] do not affect the SPE.

*Note: The application must first enable the **hsp_hclk_core_ck** (via the RCC) to change the value of CDRLCK[1:0]. Once CDRLCK[1:0] is updated, the **hsp_hclk_core_ck** can be disabled.*

If the HSP was previously activated, it is recommended to reset the HSP before reusing the shared SRAM. The HSP reset is handled by the RCC.

Table 140 shows the address offset of each memory section. The BRAM is seen as two single ports, or like a dual port (BRAM-AB). Depending on the operations requested from the HSP, the application must use a dual-port BRAM or single-port BRAMs.

Table 140. Memory mapping seen from MMC AHB slave(1)

Section start/stop acronym	Generic address offset (bytes)
BRAMAB_END	BRAMAB_START + (BRAM_SIZE * 512) - 1
BRAMAB_START	BRAMB_START + BRAM_SIZE * 512
BRAMB_START	BRAMA_START + BRAM_SIZE * 512
BRAMA_START	DRAM_START+ DRAM_SIZE * 1024
DRAM_START	CRAM_SIZE * 1024
CRAM_START	0

1. CRAM_SIZE, DRAM_SIZE, and BRAM_SIZE represent the size, in Kbytes, of CRAM, DRAM, and BRAM respectively. Refer to section *Section 20.3: HSP implementation* for memory size.

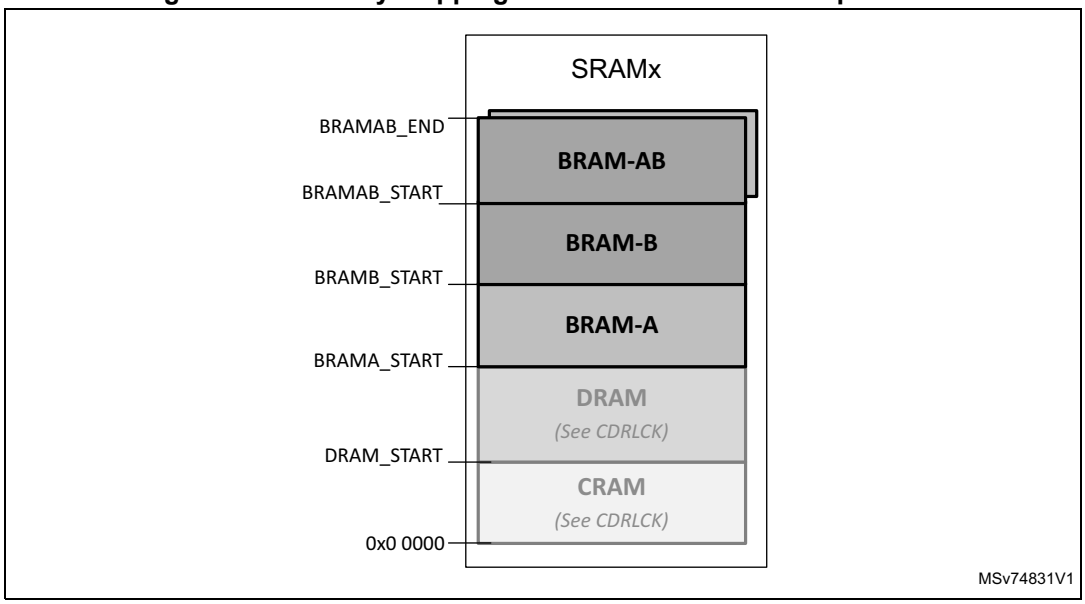
As shown in *Figure 160*, the application sees the SRAM shared with the HSP as a contiguous memory block, with the capacity to restrict access to CRAM and DRAM blocks. When the HSP is enabled, the CPU can still access this SRAM, but the HSP also uses it.

For the HSP to run safely, it is recommended to write-protect CRAM and DRAM as soon as the application has loaded the main firmware code and its data into CRAM and DRAM.

The CPU can access:

- BRAM-A, BRAM-B, and DRAM in 8, 16, or 32 bits (and in 64 bits, if the product supports it)
- CRAM in 8, 16, or 32 bits.

Figure 160. Memory mapping seen from the AHB slave port SP1



Memory resource arbitration

When the HSP is running, the BRAM-AB, BRAM-A, or BRAM-B can be accessed by the SPE or by the application through the AHB slave interface.

To keep the best SPE performances, the MMC arbiter gives the highest priority to the SPE. However, some processing done by the SPE can take 100% of the BRAM resource, preventing the other requesters from accessing the BRAM. To avoid this, the MMC ensures that if the application requests the BRAM resource, the resource is granted in the worst case after 16, 8, or 4 cycles. BARB[1:0] in the *HSP control register (HSP_CR)* allows the selection of one of these values.

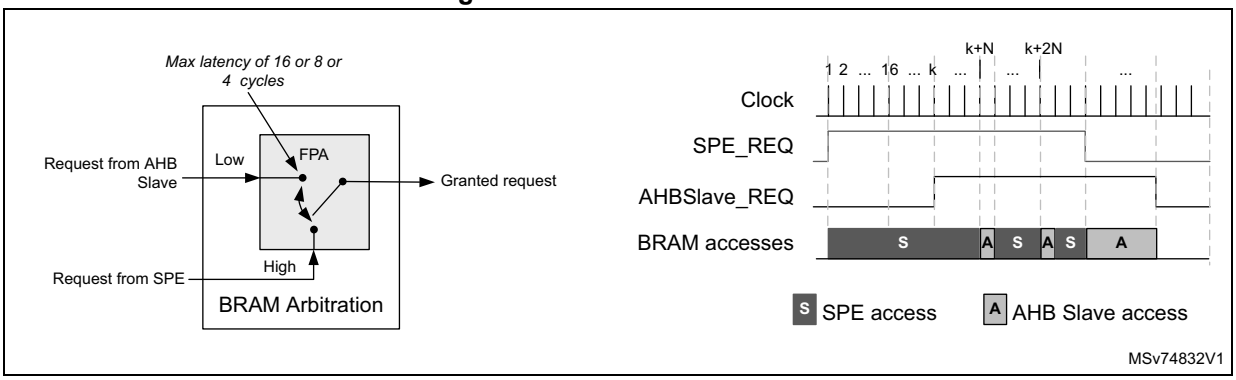
As shown in *Figure 161*, a fixed-priority arbiter (FPA) ensures that the request coming from the AHB slave interface is granted at most every 16, 8, or 4 cycles. This mechanism prevents important degradation of SPE performances.

The right part of the figure shows an arbitration example.

- The SPE occupies 100% of the BRAM bandwidth. After k clock periods, the AHB slave must access the BRAM-AB.
- After N cycles (k + N), the AHB slave can perform its transaction. N can be either 4, 8, or 16.

When the SPE no longer uses the BRAM, the AHB slave transactions are served immediately using the full bandwidth.

Figure 161. BRAM arbitration



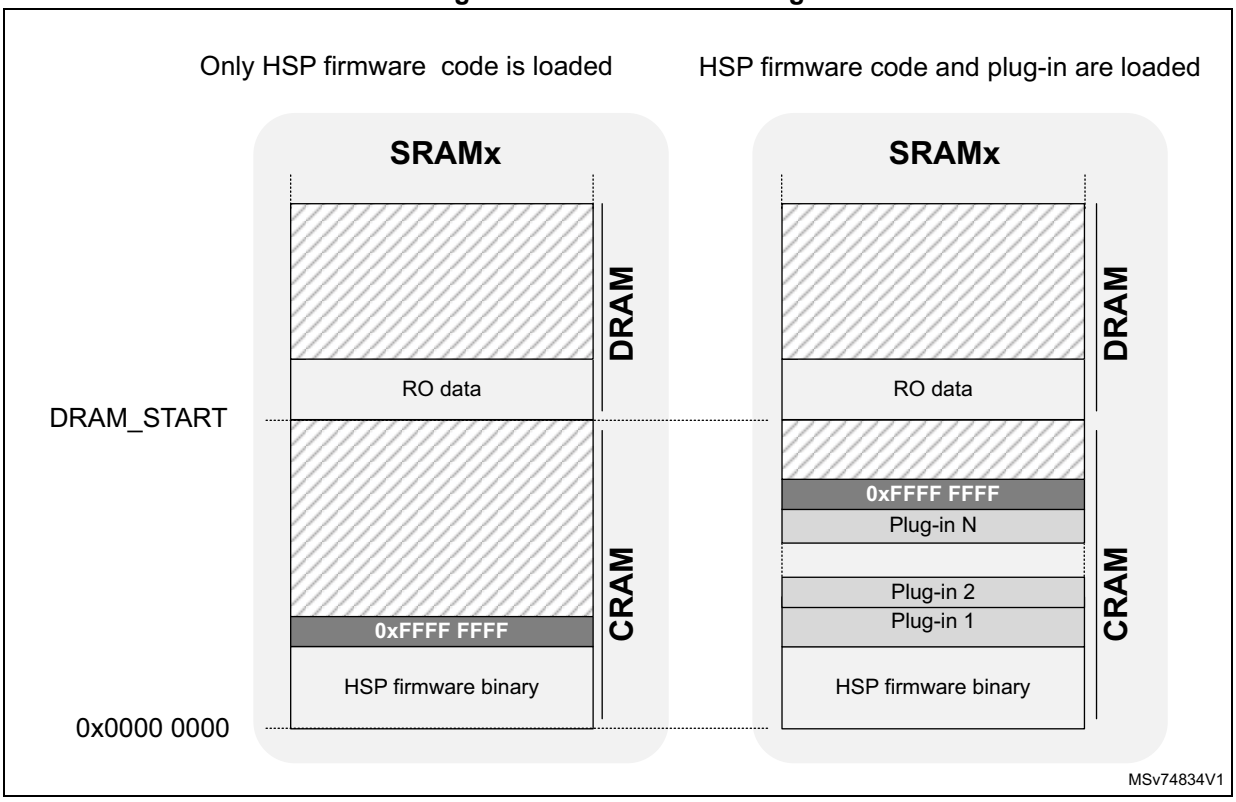
20.4.5 HSP boot

Before running the HSP, the application must copy the HSP firmware code into the code RAM (CRAM) and data RAM (DRAM). Additionally, some applications can also request to copy plug-ins. Plug-ins are extra binary codes provided by STMicroelectronics and used to correct the limitations of existing processing functions, or to add new processing functions not included in the original product.

The binary files must be copied as shown in *Figure 162*:

- If no plug-ins are loaded, the application must write a termination word (0xFFFF FFFF) right after the copied binary code.
- If plug-ins must be added, they must be copied contiguous to the HSP firmware binary code, and the termination word must be added right after the last plug-in.

Figure 162. HSP code loading



Refer to *Section 20.3: HSP implementation* to find which SRAM instance contains BRAM, CRAM, and DRAM on the product, as well as the relative address offset.

The application must execute the sequence below to run the HSP properly:

1. Reset the HSP using the reset bit in the RCC.
2. Copy the HSP firmware binary code and the RO data into the CRAM and DRAM sections respectively. Copy plug-in binaries, if any, into the CRAM.
3. Enable the HSP core clock via the RCC.
4. If needed, protect CRAM and DRAM sections against accidental corruption via CDRLCK[1:0].
5. Set the BOOTEN bit to run the HSP. The HSP executes the code starting at the CRAM base address of the CRAM, and enters the HSPCFG state.

6. In HSPCFG state, the HSP firmware performs the following operations:
   a) Initialize the internal HSP subblocks. TSKLUT is also initialized to only allow the activation of the supervisor (PLSUP) or Direct Command Accelerator functions (PLDCMD).
   b) Copy the firmware versions and SPE into HSP_FWVERR and HSP_SPEVERR.
   c) Set BSTAT[3:0] to 1, indicating it is waiting for the FW_INIT command.
7. The HSP firmware is now ready to accept the FW_INIT command.
8. When the FW_INIT command is complete, the HSP firmware activates the plugin codes, if any. This phase sets BSTAT[3:0] to 8.
9. The HSP enters the RUN phase and the WFE state, and sets BSTAT[3:0] to 2. As soon as an enabled event is activated, the HSP exits the WFE state and goes to:
   a) PLRUN if the selected event is valid (meaning it is associated with an existing processing list). The HSP firmware executes the processing list corresponding to the selected event, and goes back to WFE.
   b) PLSUP if the selected event corresponds to the supervisor. The HSP firmware executes the command sent by the application and goes back to WFE. Processing lists are created in this mode.
   c) PLDCMD if a direct command is programmed via the DCMD block.
   d) PLFAIL if the selected event is not valid.

   In PLDCMD mode, the HSP firmware executes the command sent by the application and goes back to WFE.

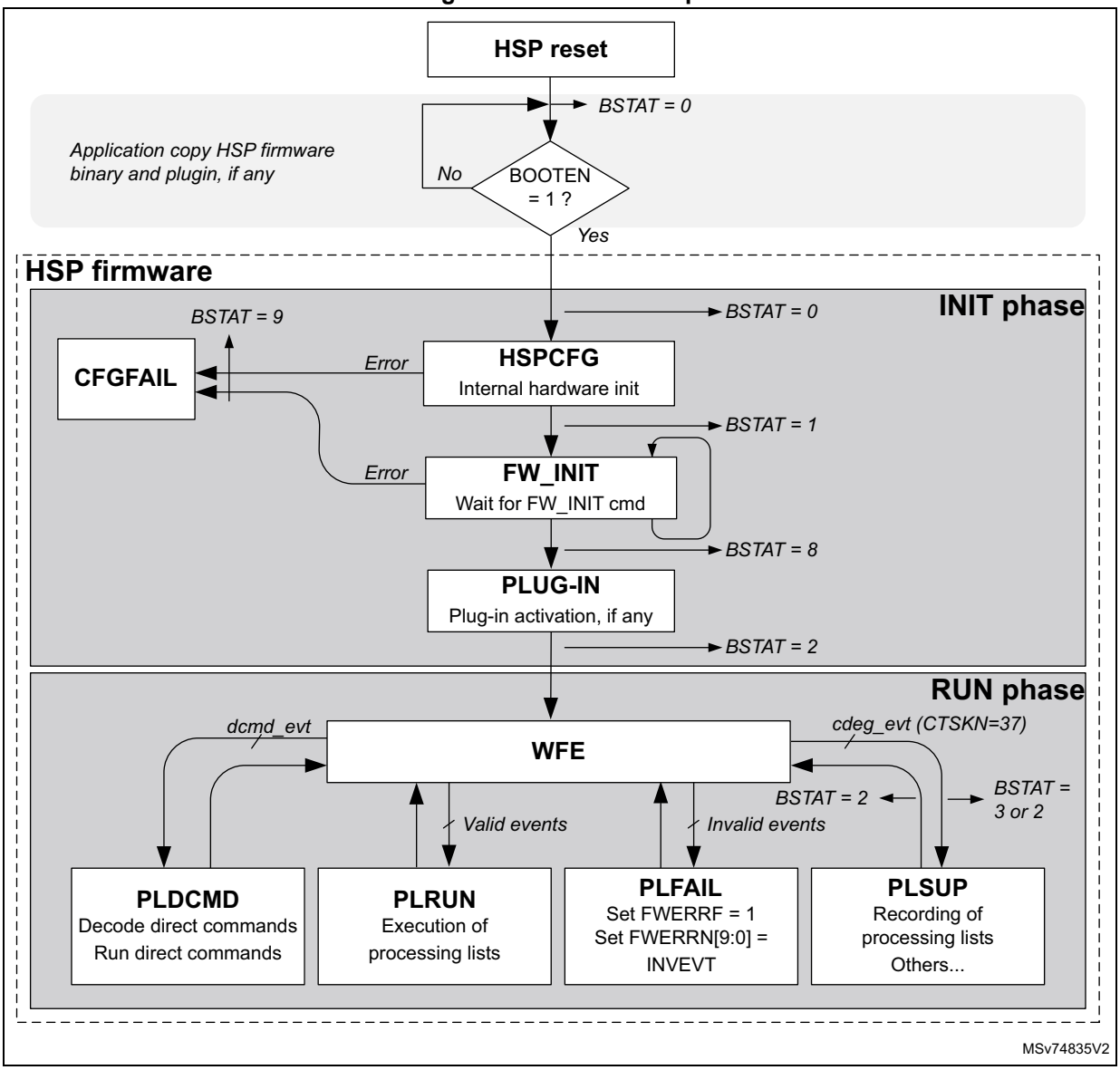
   In PLFAIL, the HSP firmware sets the FWERRF bit, sets the firmware error code in FWERRN[9:0], and goes back to WFE.

If an error occurs during this initialization process, the HSP firmware goes to the CFGFAIL state. In the CFGFAIL state, BSTAT[3:0] is set to 9. To set the HSP firmware back to the HSPCFG state, the HSP must be reset through the RCC.

When the HSP firmware initialization is completed, the application can enable the events, which are supposed to trigger the programmed processing list.

BOOTEN, CDRLCK[1:0], and BSTAT[3:0] are located in the *HSP control register (HSP_CR)*.

Figure 163. HSP start-up



The table below shows the different values BSTAT can take.

Table 141. BSTAT possible values

Conditions	BSTAT[3:0]
Information	
Default value after reset	0b0000
The HSP firmware started	0b0010
The HSP firmware waits for firmware initialization command (FW_INIT)	0b0001
The HSP firmware is in RUN phase	0b0011
The HSP firmware is recording a processing list	0b0100
Errors	
The HSP firmware is in CFGFAIL state	0b1100

20.4.6 Message box (MSGB)

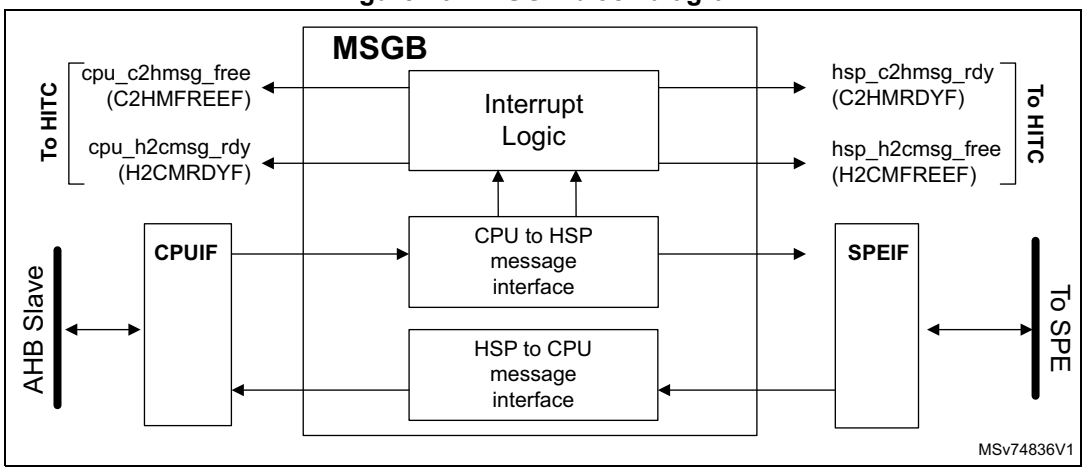
The message box (MSGB) offers various services dedicated to the HSP firmware configuration.

The MSGB block contains an HSP-to-CPU and a CPU-to-HSP message interface with:

- Synchronization flags and interrupt capabilities
- Two payload registers

The MSGB is generally used with the *HSP parameter register x (HSP_PARAMRx)* to provide the message payload. Messages are used to configure the HSP firmware part, especially to record the processing list.

Figure 164. MSGB block diagram



CPU to HSP message interface

Using polling mode

When the CPU wants to send a message to the HSP, it has to do the following:

1. Read C2HSEM in the *HSP CPU-to-HSP semaphore register (HSP_C2HSEMR)*.
2. Wait until C2HSEM = 0, indicating that the previous message has been read by the HSP and a new message can be transferred.
3. Write a new value in the *HSP CPU-to-HSP message data register (HSP_C2HMSGDR)*.
4. Set C2HSEM.

On its side, the HSP does the following:

1. Read the C2HSEM bit.
2. Wait until C2HSEM = 1, indicating that a new message is available.
3. Read the C2HDATA.
4. Clear C2HSEM to indicate that the message has been read.

Using interrupt mode

The MSGB can generate interrupts to the CPU when the CPU-to-HSP mailbox is ready to receive a new message (through the C2HMFREEF flag).

The CPU must first enable the interrupts by executing the sequence below:

1. Clear the C2HMFREEF flag by setting C2HMFREEC in the *HSP events interrupt clear register (HSP_EVT_ICR)*.
2. Allow the C2HMFREEF flag to generate an interrupt by setting C2HMFREEIE in the *HSP events interrupt enable register (HSP_EVT_IER)*.

When an event interrupt occurs, the interrupt subroutine (ISR) must do the following:

1. Read the C2HMFREEF flag in the *HSP events interrupt status register (HSP_EVT_ISR)*.
2. If C2HMFREEF is set:
   - The CPU can clear this flag by setting C2HMFREEC.
   - The CPU can write a new message.
   - When the CPU wants to notify the HSP that a new message is available, C2HSEM must be set.

   The last two steps can be done outside the ISR.

HSP-to-CPU message interface

Using polling mode

When the HSP wants to send a message to the CPU, it does the following:

1. Read the H2CSEM flag.
2. Wait until H2CSEM = 0, indicating that the previous message has been read by the CPU and the content of HSP_SPE_H2CMSGDR can be overwritten.
3. Write the new value into C2HDATA.
4. Set H2CSEM.

On its side, the CPU does the following:

1. Read H2CSEM in the *HSP HSP-to-CPU semaphore register (HSP_H2CSEMR)*.
2. Wait until H2CSEM = 1, indicating that a new message is available.
3. Read the *HSP HSP-to-CPU message data register (HSP_H2CMSGDR)*.
4. Clear H2CSEM in the *HSP HSP-to-CPU semaphore register (HSP_H2CSEMR)* to indicate that the message has been read.

Note: Handling the MSGB in polling mode also sets H2CMRDYF and C2HMFREEF. Prior to using these flags, the software must clear them.

If the CPU and the HSP set and clear C2HSEM at the same time respectively, C2HSEM is cleared.

If the CPU and the HSP set and clear C2HSEM at the same time respectively, C2HSEM is cleared.

Using interrupt mode

The MSGB can generate interrupts to the CPU when a new message is available for the CPU (through the H2CMRDYF flag).

The CPU must first enable the interrupts by executing the sequence below:

1. Clear the H2CMRDYF flag by setting the H2CMRDYC bit in the *HSP events interrupt clear register (HSP_EVT_ICR)*.
2. Allow the H2CMRDYF flag to generate an interrupt by setting the H2CMRDYIE bit in the *HSP events interrupt enable register (HSP_EVT_IER)*.

When an event interrupt occurs, the interrupt subroutine (ISR) must do the following:

1. Read the H2CMRDYF flag in the *HSP events interrupt status register (HSP_EVT_ISR)*.
2. If H2CMRDYF is set:
   - The CPU can clear this flag by setting H2CMRDYC.
   - The CPU can read the message.
   - When the CPU decides to free the message box, the H2CSEM bit must be cleared.

   The last two steps can be done outside the ISR.

20.4.7 Direct command interface (DCMD)

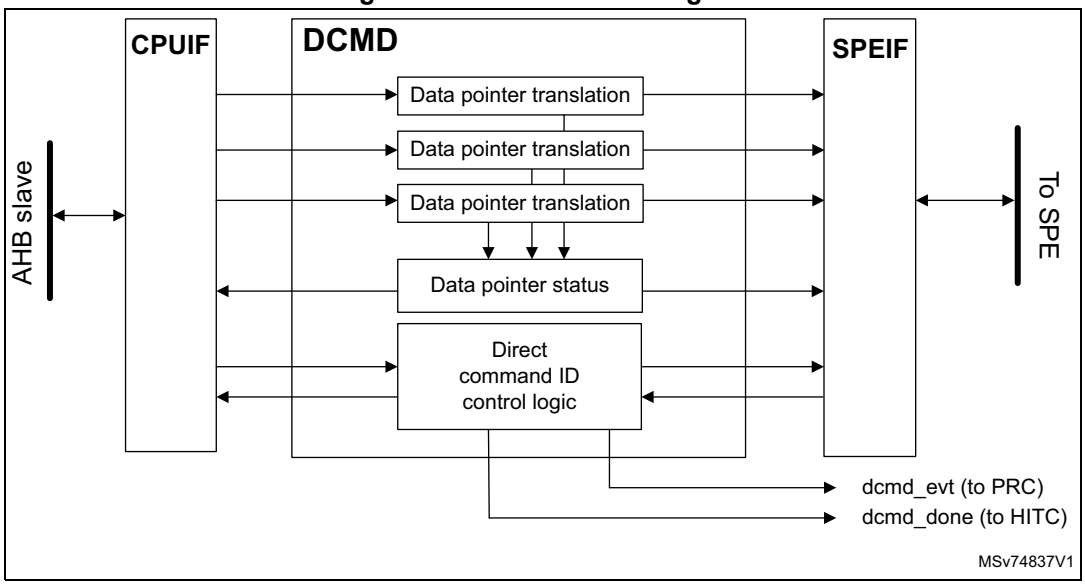
The direct command interface (DCMD) is dedicated to the execution of direct commands, reducing the command overhead in an efficient way. It is used for HSP accelerator mode. This block is controlled by the HSP with a PLDCMD dedicated processing list (refer to *Section 20.6.4: Executing direct commands (accelerator mode)*).

The DCMCD provides the following services:

- Direct command ID register, for triggering events
- Up to three data pointer interfaces
- Status flags

The DCMD is generally used with the *HSP parameter register x (HSP_PARAMRx)* to provide the parameters corresponding to the command that the application wants to execute.

Figure 165. DCMD block diagram



Prior to running a direct command, the SPE must perform the following preliminary operations:

1. Decode the command
2. Adapt the data pointer values
3. Execute the command

To reduce latency in command decoding, the DCMD block allows the SPE to start command decoding while the CPU is still providing the parameters.

By default, the DCMD block is enabled. DCMDDIS in the *HSP BUFF configuration register (HSP_BUFFCFGR)* controls the enabling and disabling of this function. This bit is write-once after an HSP reset. If the application wants to ensure that the DCMD cannot be accidentally disabled, it must clear DCMDDIS. If the application wants to disable the DCMD, DCMDDIS must be set.

The CPU can transmit the command ID to be executed by writing the command ID in the *HSP DCMD command ID register (HSP_DCMDIDR)*.

A write access to this register triggers two actions:

- It sets the DCBSY flag.
- It asserts the DCMD event (**dcmd_evt**).

dcmd_evt triggers a processing list dedicated to the execution of direct commands. The number of this processing list is 38, and its priority is set to 26.

The DCBSY flag is cleared by the SPE when the direct command execution is completed.

The *HSP DCMD address pointer register x (HSP_DCMDPTRx)* allows the CPU to provide to the SPE the address pointers requested by the direct commands.

When the CPU writes an address pointer into one of the HSP_DCMDPTRx registers, the HSP hardware sets the corresponding PTRF[x] flag in the *HSP DCMD pointer status register (HSP_DCMDPTSR)*.

When the SPE reads this pointer, the DCMD hardware sets the corresponding PTRFx bit to zero. The DCMD also provides a bit (RDOG) indicating if the SPE is performing a read operation of one of the three data pointers. This bit can be used in a debug phase to check if the SPE is waiting for a data pointer value.

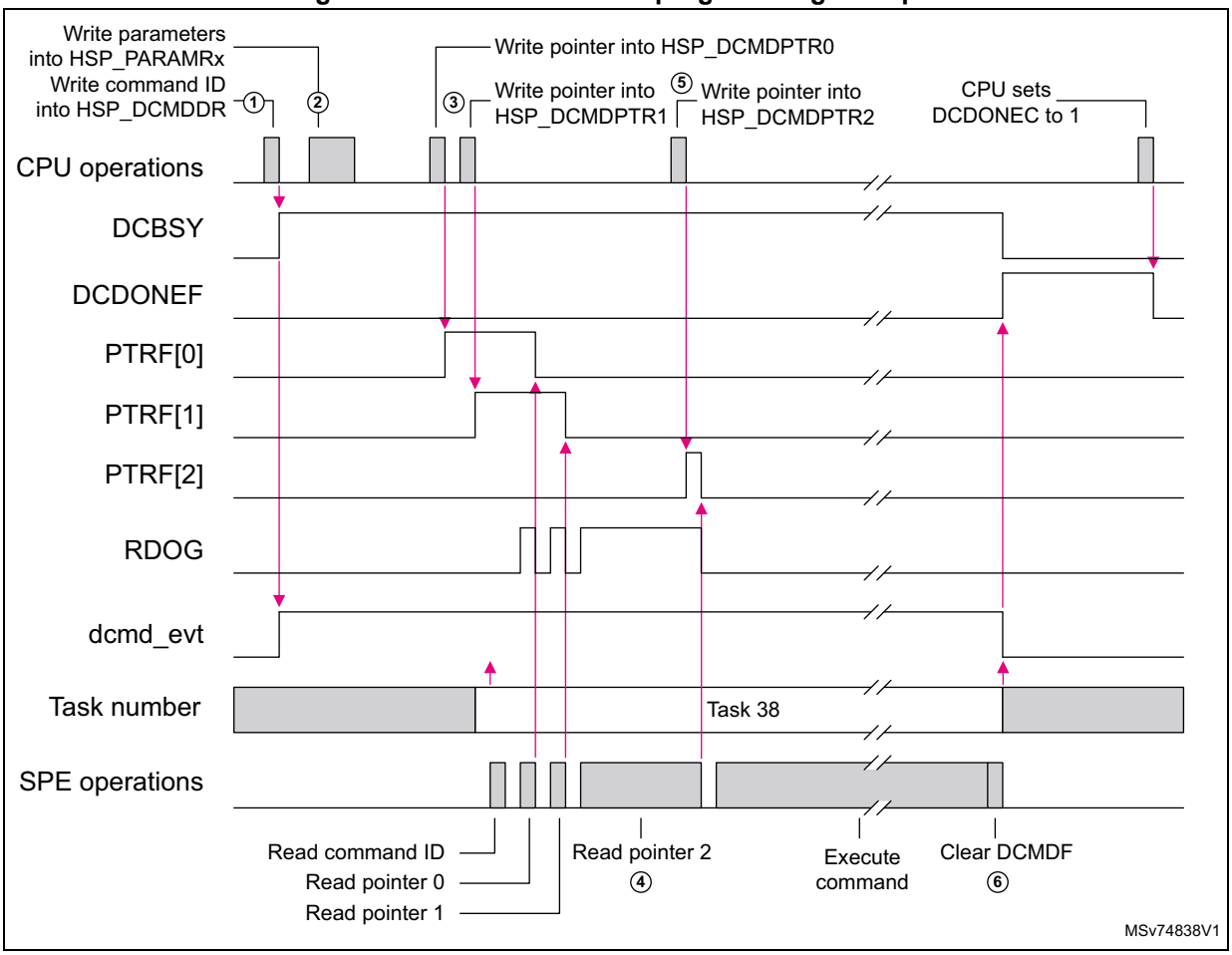
Note: If the SPE reads a pointer register not yet updated by the CPU, the SPE remains frozen on the read operation until the CPU has updated the corresponding pointer register.

The DCMD also offers the possibility to generate an interrupt when the DCBSY flag is cleared, meaning that the command has been completed. The interrupt is enabled by setting DCDONEIE in the *HSP events interrupt enable register (HSP_EVT_IER)*.

Figure 166 shows a typical example of a command programming sequence, and the behavior of the various signals. During an initialization sequence, the CPU is assumed to clear DCMDDIS.

1. The CPU writes the command ID in the *HSP DCMD command ID register (HSP_DCMDIDR)*. The DCBSY and dcmd_evt are set (#1 in the figure).
2. The CPU writes data parameters in the *HSP parameter register x (HSP_PARAMRx)* (#2 in the figure).
3. The CPU starts to write the first two address pointers in the *HSP DCMD address pointer register x (HSP_DCMDPTRx)*. As soon as the EVTC allows it, task number 26 is activated (#3 in the figure).
4. The SPE first reads the command ID to check which command must be executed, and then starts to read the necessary address pointers. The first two address pointers are already available, but the SPE remains blocked for several cycles when reading the third pointer, because the CPU did not yet write it (#4 in the figure).
5. When the CPU writes the third pointer, the SPE gets it translated, and the command can be executed (#5 in the figure).
6. When the command execution is completed, the SPE clears the DCBSY flag. This action clears the dcmd_evt (#6 in the figure).
7. Finally, if DCDONEIE is set, an interrupt can be generated to inform that task execution has been completed. Then the CPU can clear the interrupt (and DCDONEF flag) by setting the clear bit (DCDONEC).

Figure 166. Direct command programming example



If the SPE reads a data pointer not written by the CPU, the SPE remains blocked in the reading operation. The SPE can no longer serve interrupts, nor enter debug mode. The SPE can exit this freeze state through one of the following actions:

- The application provides the expected data pointer.
- DCMDDIS is set (if possible). The SPE unfreezes, but executes a command with an invalid pointer, which could be unsafe.
- The HSP is reset.

The HSP also provides a signal event for the CPU named **hsp_txev,** indicating that the current direct command execution has been completed. Every time a direct command execution is completed, a single-cycle pulse is generated. DCMDEVEN in the *HSP control register (HSP_CR)* allows this function activation (see also *Section 20.4.12: HSP interrupt controller (HITC)*).

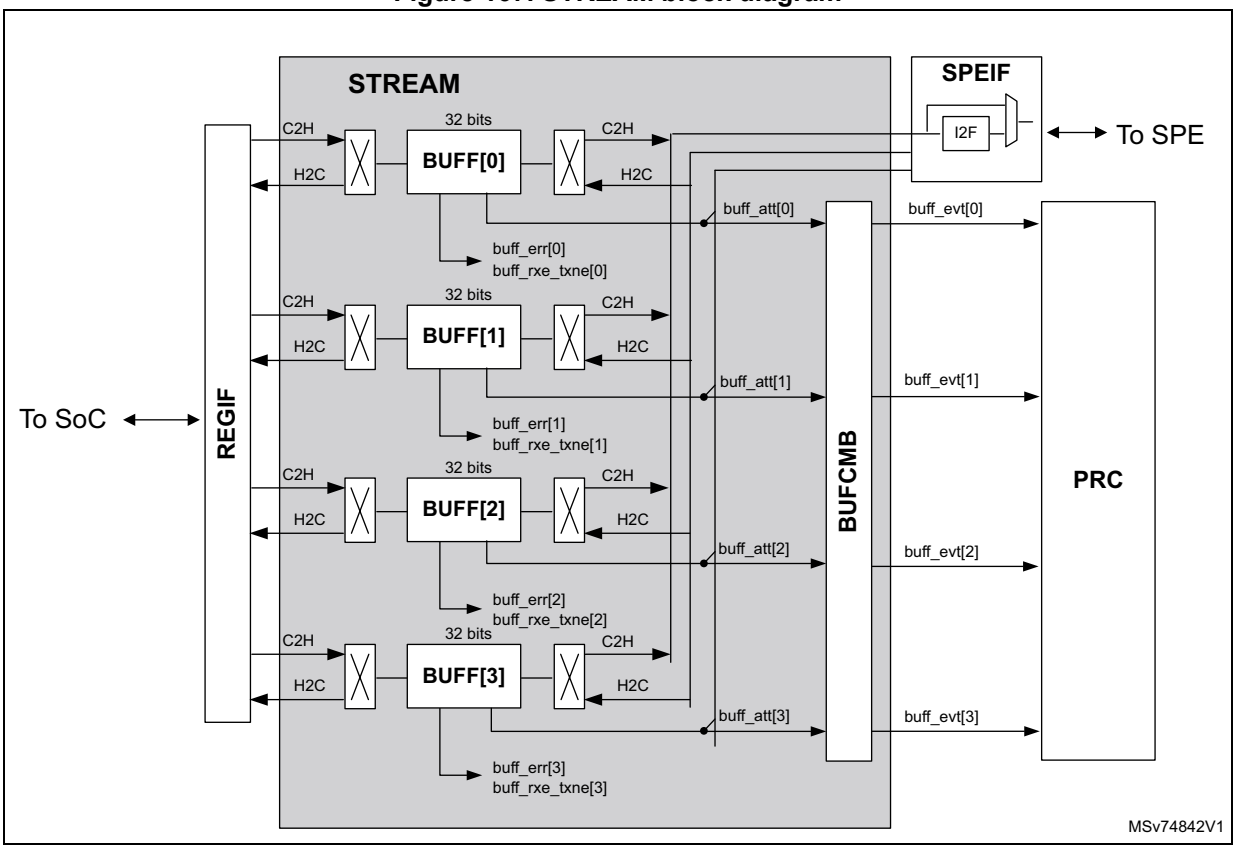
20.4.8 STREAM interface

The STREAM interface can be used if the application wants to trigger an HSP processing list when data are sent to or read from the STREAM interface. It allows the application to set up source- or sink-driven processing.

For example, a DAC associated with a DMA device can take its data from the STREAM interface. As soon as the STREAM interface is read by the DMA device, the SPE is advised and the associated processing list, recorded by the application during an initialization phase, is executed immediately. Typically, this processing list prepares the next sample for the DAC.

Conversely, an ADC associated with a DMA device can provide its data to the STREAM interface. As soon as the STREAM interface is written by the DMA device, the SPE is advised and the associated processing list, recorded by the application during an initialization phase, is executed immediately. Typically, this processing list processes the incoming sample and frees the STREAM interface for the next data.

Figure 167. STREAM block diagram



The STREAM interface provides up to four data buffers of 32 bits. Each of them can be configured either in the CPU-to-HSP direction (C2H) or the HSP-to-CPU direction (H2C). Buffers configured in CPU-to-HSP direction are called C2HBUFF in this document, and buffers configured in HSP-to-CPU direction are called H2CBUFF.

These buffers can be accessed by the application through the *HSP BUFFx data register x (HSP_BUFFxDR)*.

The data provided by the CPU (C2H direction) can be converted to float32 automatically if the I2FEN bit in the *HSP BUFF configuration register (HSP_BUFFCFGR)* is set.

The conversion from integer to float32 only works for the following input formats:

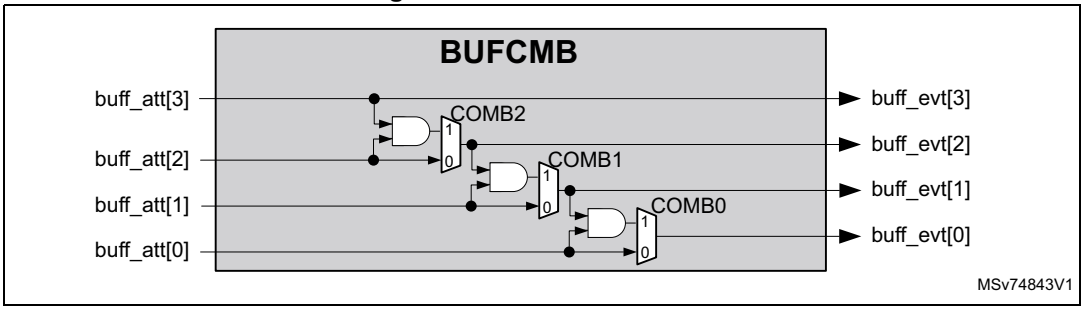
- Signed integers up to 32 bits. The sign extension must always be performed up to 32 bits. The accuracy of a signed 32-bit integer is reduced to 24 bits due to the float32 format.
- Unsigned integers up to 31 bits. Bit 31 must be cleared. The accuracy is limited to 24 bits due to the float32 format.

The STREAM buffers configured in the C2H direction assert the **buff_att** signal when a data is written into a STREAM buffer by the application. The **buff_att** signal remains active as long as the SPE does not read the corresponding STREAM buffer.

The STREAM buffers configured in the H2C direction assert the **buff_att** signal when they are empty. The **buff_att** event signal remains active as long as the SPE does not write data into the corresponding STREAM buffer.

The BUFCMB block adds additional flexibility by combining the **buff_att**[x] signals to generate the buffer events (**buff_evt**[x]). The BUFCMB allows the generation of the **buff_evt**[x] event, when the corresponding **buff_att**[x] signal is activated or when both **buff_att**[x] and **buff_att**[x+1] signals are activated. The BUFCMB is configured in the *HSP BUFF configuration register (HSP_BUFFCFGR)*.

Figure 168. BUFCMB details



The table below lists some of the possible configurations:

Table 142. BUFFCMB programming examples

COMB2	COMB1	COMB0	-	Outputs
0	0	0	→	**buff_evt**[x] generated when **buff_att**[x] is activated
1	0	1	→	**buff_evt**[0] generated when **buff_att**[0] & **buff_att**[1] are activated **buff_evt**[1] generated when **buff_att**[1] is activated **buff_evt**[2] generated when **buff_att**[2] & **buff_att**[3] are activated **buff_evt**[3] generated when **buff_att**[3] is activated

Table 142. BUFFCMB programming examples (continued)

COMB2	COMB1	COMB0	-	Outputs
0	1	1	→	**buff_evt**[0] generated when **buff_att**[0] & **buff_att**[1] & **buff_att**[2] are activated **buff_evt**[1] generated when **buff_att**[1] & **buff_att**[2] are activated **buff_evt**[2] generated when **buff_att**[2] is activated **buff_evt**[3] generated when **buff_att**[3] is activated
1	1	1	→	**buff_evt**[0] generated when **buff_att**[0] & **buff_att**[1] &...& **buff_att**[3] are activated **buff_evt**[1] generated when **buff_att**[1] & **buff_att**[2] & **buff_att**[3] are activated **buff_evt**[2] generated when **buff_att**[2] & **buff_att**[3] are activated **buff_evt**[3] generated when **buff_att**[3] is activated

The CPU can configure the buffer directions and the BUFCMB with the *HSP BUFF configuration register (HSP_BUFFCFGR)*.

The STREAM interface is activated by setting STREAMEN in the *HSP BUFF configuration register (HSP_BUFFCFGR)*. When STREAMEN is set, it is no longer possible to change the buffer direction, nor to modify the BUFFCMB configuration.

Note: Clearing STREAMEN resets the STREAM block, and all pending buffer events are cleared. Any write access operation to the STREAM buffers is ignored, and read access returns the last data.

The STREAM interface generates an error signal for each buffer and the error signals **buff_err**[3:0] are asserted:

- If new data are written into a buffer configured in the C2H direction while the previous data have not been read by the SPE, an overrun error is detected. The new data are written, and the previous ones are lost.
- If a buffer configured in the H2C direction is read by the CPU while the SPE did not have time to write the new data, an underrun error is detected. In this case, the CPU reads the previous data.

These error signals activate the B[3:0]ERRF flags in the *HSP error interrupt status register (HSP_ERR_ISR)*. An interrupt to the CPU can be generated if the corresponding interrupt enable flag is set.

The STREAM interface also generates buffer status signals (**buff_rxe_txne**[3:0]). These signals set the corresponding B[3:0]EVTF in the *HSP events interrupt status register (HSP_EVT_ISR)* in each of the following conditions:

- If the corresponding C2H buffer is empty
- If the corresponding H2C buffer is not empty

An interrupt to the CPU can be generated if the corresponding interrupt enable flag is set.

buff_att[x] signals are also provided to the TRGO to generate a trigger pulse on **hsp_trgo**[x] if the SPE reads a C2H buffer, or writes into the H2C buffer.

Figure 169 shows how buff_evt[x], BxEVTF, buff_evt[y], ByEVTF, and TRGO (when enabled) behave in the case of C2H and H2C buffers, when BUFCMB is configured with its

default values (bypass). When STREAMEN is cleared, the complete STREAM interface is reset and the **buff_evt**[x] and **buff_rxe_txne**[x] signals return to 0.

Case where BUFF[x] is set in the C2H direction:

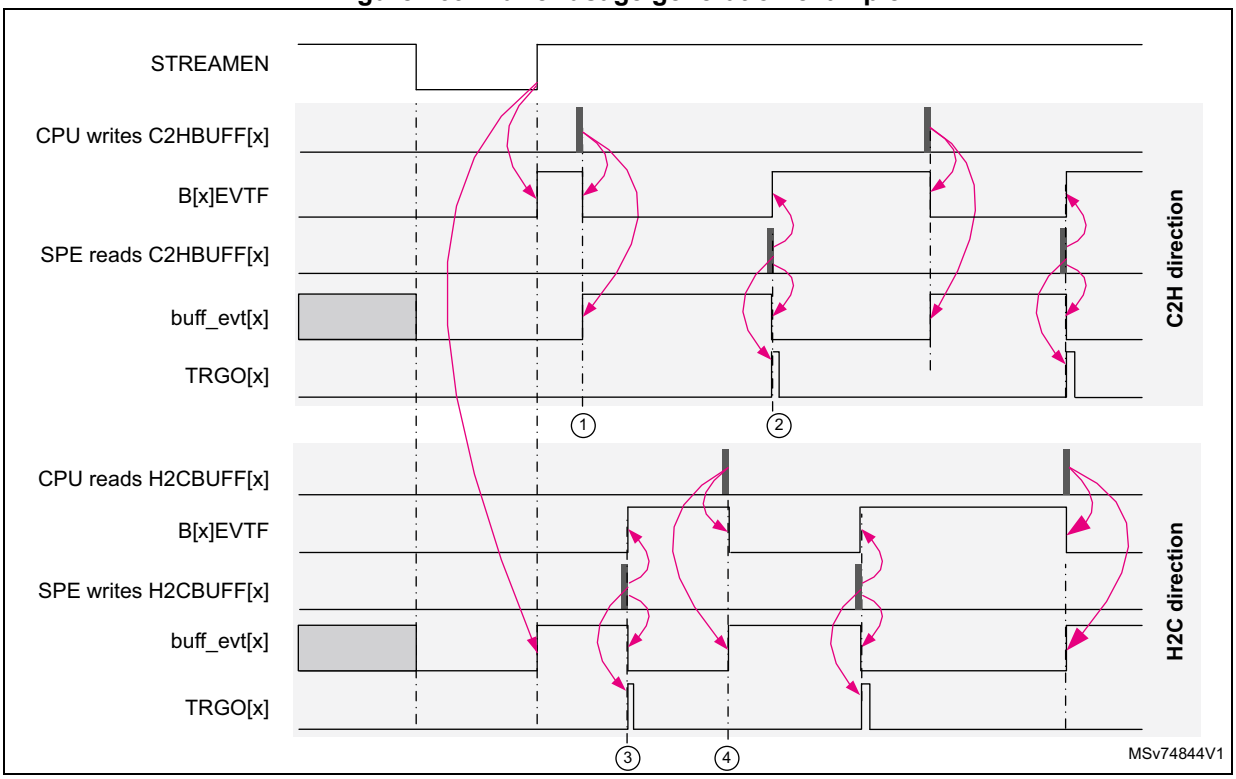
1. As soon as STREAMEN is set, B[x]EVTF goes high, indicating that BUFF[x] is empty.
2. When the CPU writes into a BUFF[x], the B[x]EVTF flag goes back to 0 and **buff_evt**[x] is activated, requesting SPE processing (#1 in the figure).
3. When the SPE reads BUFF[x], the B[x]EVTF flag is set again, **buff_evt**[x] is released, and a TRGO[x] pulse is generated if selected for the STREAM buffer (#2 in the figure).

Case where BUFF[x] is set in the H2C direction:

1. As soon as STREAMEN is set, the **buff_evt**[y] signal goes high, requesting SPE activation.
2. When the SPE writes BUFF[x], the B[x]EVTF flag is set, **buff_evt**[x] is released, and a TRGO[y] pulse is generated if selected for the STREAM buffer (#3 in the figure).
3. When the CPU reads BUFF[x], the B[x]EVTF flag goes back to 0 and **buff_evt**[x] is activated, requesting SPE processing (#4 in the figure).

When the SPE writes into a BUFF[x], the **buff_evt**[y] flag goes back to 0, and a TRGO[x] pulse is generated if it is selected for the STREAM buffer (#1 in the figure).

Figure 169. Buffer usage generation example



20.4.9 Trigger input interface (TRGITF)

The HSP embeds up to 10 trigger input blocks, named TRGINx. Each TRGINx receives a trigger input signal from TRGMX. For each TRGINx, the application can select any trigger input signal, and program the activation of an event on a rising or falling edge of the selected signal.

TRGINx blocks can be configured through the *HSP TRGIN configuration register (HSP_TRGINCFGR)*, *HSP TRGIN input selection register 0 (HSP_TRGINSELR0)*, *HSP TRGIN input selection register 1 (HSP_TRGINSELR1)*, and *HSP TRGIN input selection register 2 (HSP_TRGINSELR2)*.

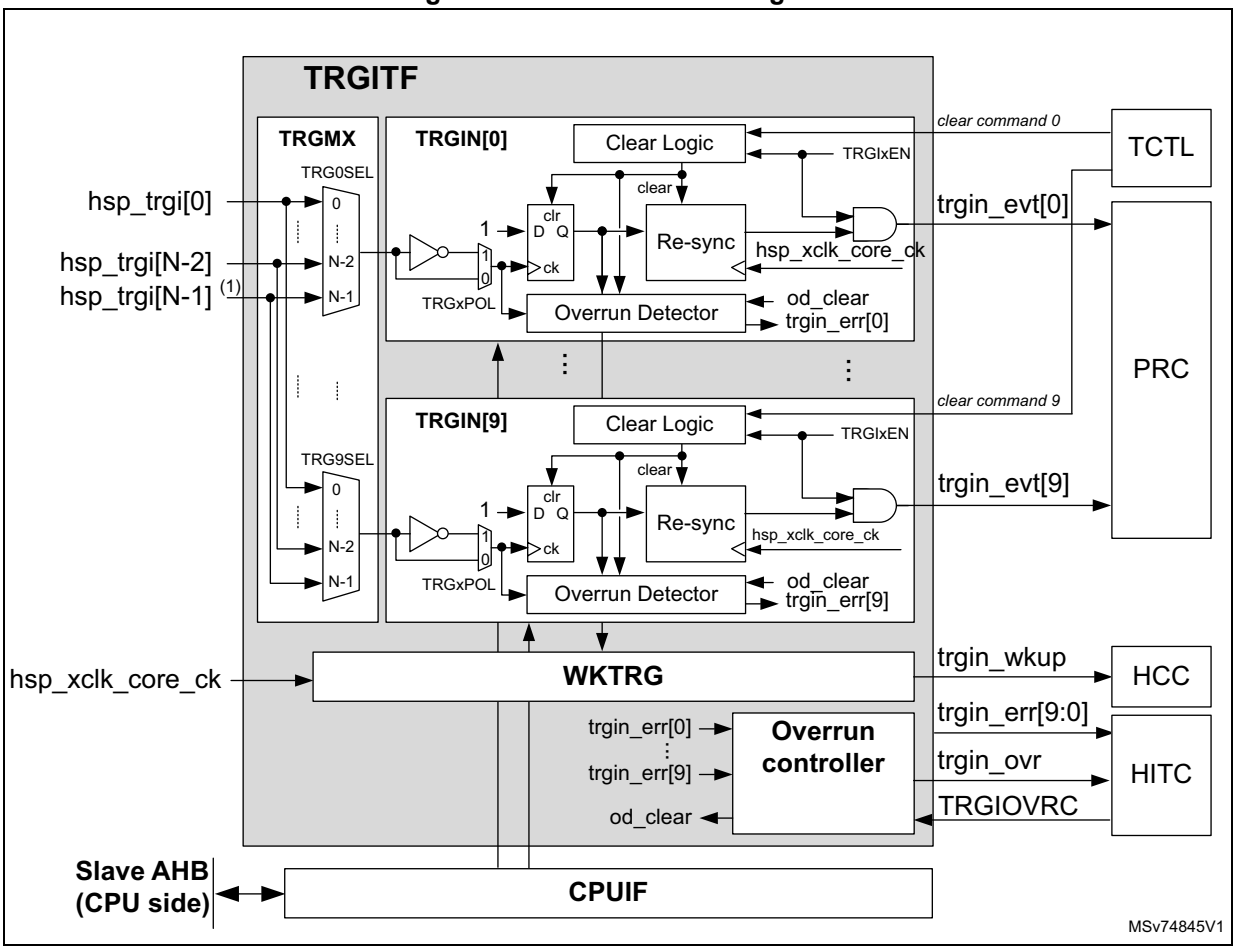
Each trigger interface can be independently enabled through the *HSP BUFF configuration register (HSP_BUFFCFGR)*.

The TRGIN blocks perform the following functions:

- Selecting the rising or falling edge sensitivity to generate an event.
- Storing a transition, even if this transition is faster than the EVTC internal clock.
- Performing the clock domain crossing.
- Storing the event until it is taken into account by the priority encoder.

The pending events are automatically cleared by the TCTL block when the SPE starts the corresponding processing.

Figure 170. TRGITF block diagram



1. The number of trigger inputs depends on the product (see *Section 20.3: HSP implementation*).

The TRGMX block allows the application to route any trigger input to any TRGIN interface.

The WKTRG block allows the detection of an enabled pending event even if the TRGITF is not clocked. This signal is used for autonomous functions or to wake up the product if needed.

Note: When TRGIxEN are cleared, the corresponding TRGIN logic is maintained to reset, avoiding the storage of a spurious transition, for example when the polarity is changed. The overrun detector and pending event are also reset when TRGIxEN is cleared.

It is recommended to change the polarity bit and the signal source selection when the corresponding TRGIxEN bit is cleared.

TRGIN overrun detection

Each TRGIN block has an overrun detector checking if a new trigger event occurred while the previous one was not yet taken into account by the SPE. The overrun detector sets the **trgin_err**[x] signal when an overrun is detected. The overrun controller generates the **trgin_ovr** signal, setting the TRGIOVRF flag. If the TRGIOVRIE bit is set, an interrupt is generated to the CPU. The application can check which TRGIN block causes the overrun by reading TRGIERR[9:0] in the *HSP error information register (HSP_ERRINFR)*. When the

application clears the TRGIOVRF flag through the TRGIOVRC bit, all overrun detectors are reset.

See *Section 20.4.12: HSP interrupt controller (HITC)*.

*Note: The overrun function does not work when **hsp_hclk_core_ck** is not present.*

Figure 171 shows a case of overrun detection, using two TRGIN inputs. In the example, **trgin_evt**[2] activates the task number 2, and **trgin_evt**[5] activates the task number 5.

hsp_trgi[2] and **hsp_trgi**[5] receive an event at the same time (#1 in the figure). **trgin_evt**[2] and **trgin_evt**[5] are activated. In the example, a higher priority is given to **trgin_evt**[2], which is executed as soon as the SPE is available. **tsk_active**[5:0] indicates that task number 2 is in execution.

While the **trgin_evt**[5] event is pending, a second trigger is detected in the **hsp_trgi**[5] input, causing an overrun error (#2 in the figure). The flag TRGIOVRF is set, and **trgin_err**[9:0] is set to 0x20 (bit 5 set). An interrupt can be activated for the application.

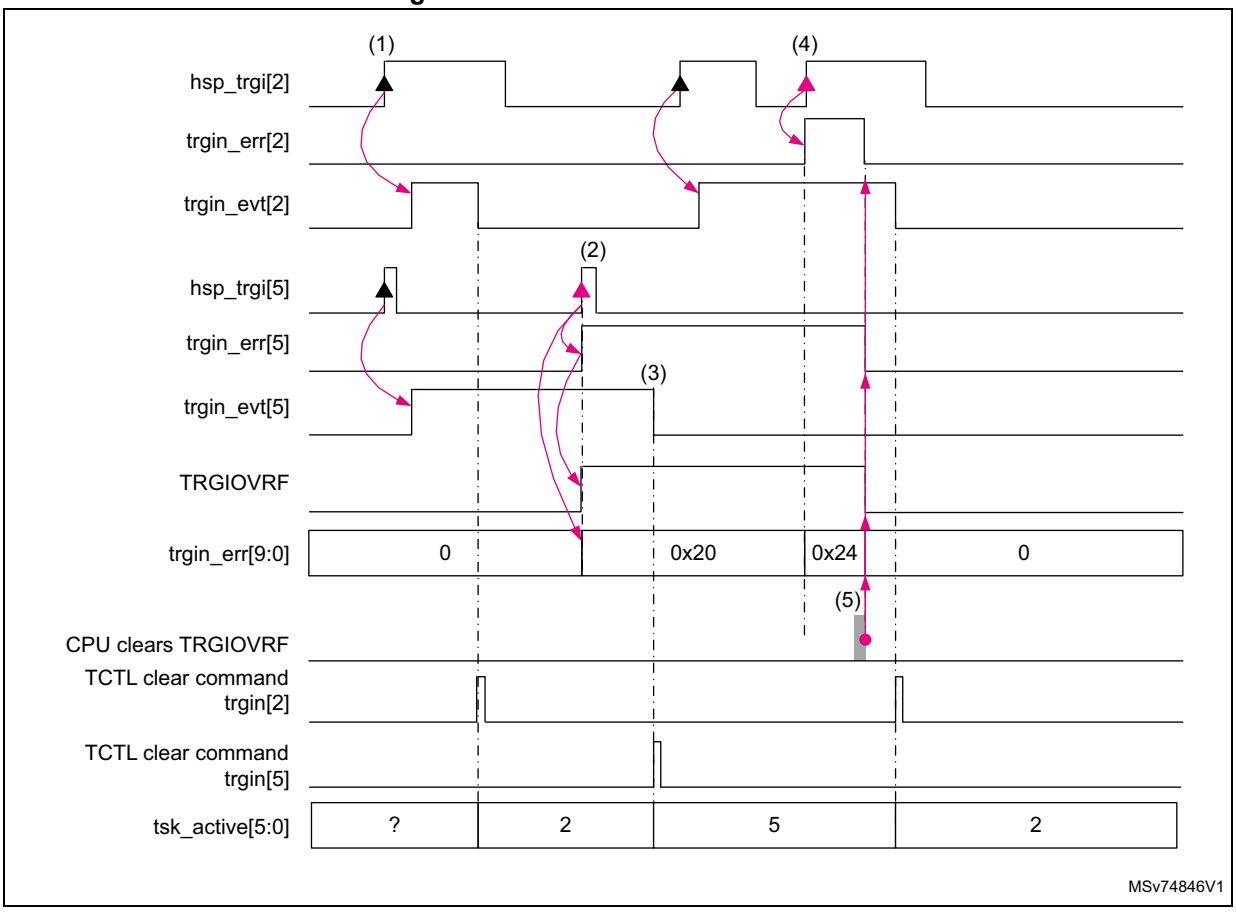
The SPE completed task number 2, and the HSP can serve the pending request from **trgin_evt**[5]. Task 5 is activated (#3 in the figure).

The **hsp_trgi**[2] receives two new triggers, the first one activates **trgin_evt**[2] and the second one activates the overrun error (#4 in the figure). The TRGIOVRF was already set, and the **trgin_err**[9:0] value is updated to 0x24 (bits 5 and 2 set).

The application reads **trgin_err**[9:0] and clears TRGIOVRF (#5 in the figure). **trgin_err**[x], TRGIOVRF, and **trgin_err**[9:0] are reset. When the SPE completes task 5, the pending event is served and task 2 is executed again.

Note: Two events have been lost due to overrun.

Figure 171. TRGIN overrun detection



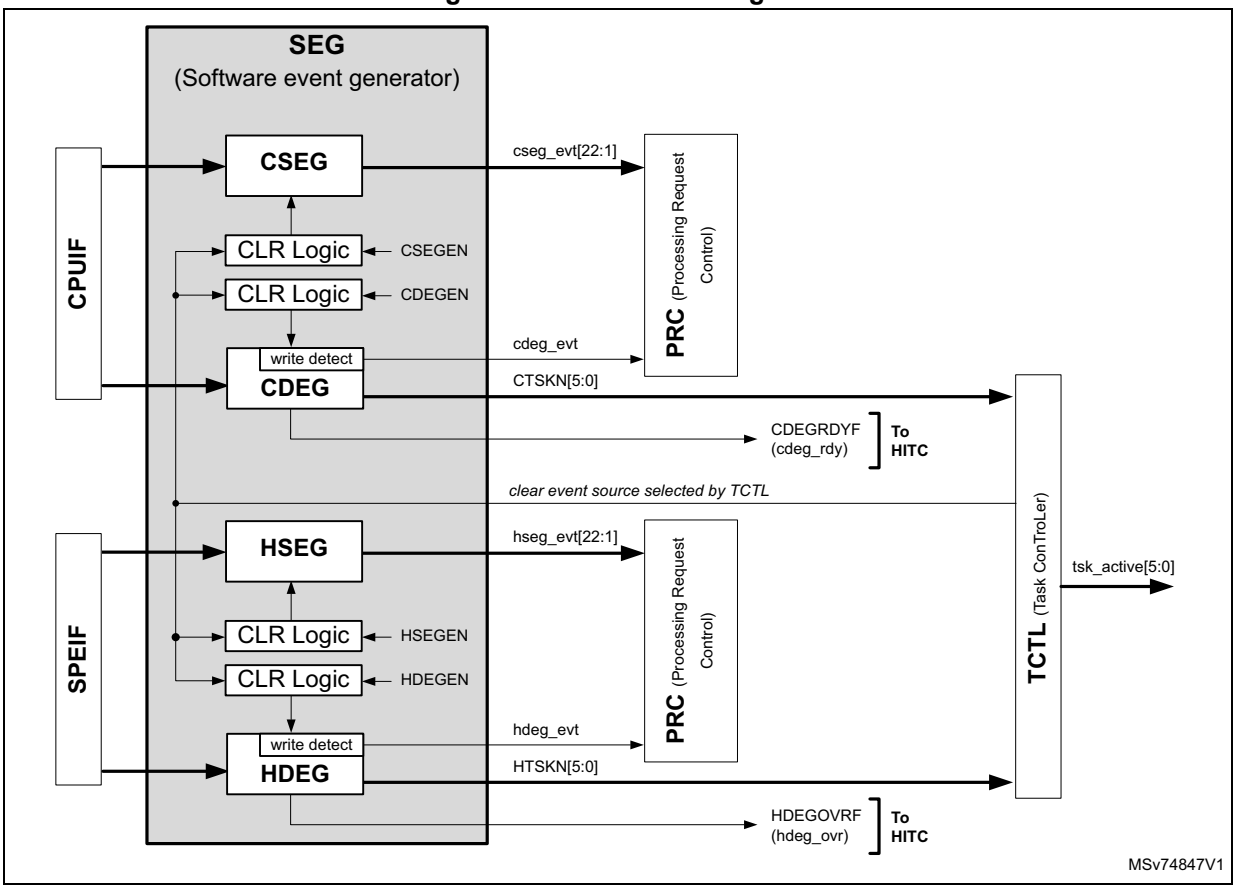
20.4.10 Software events generator (SEG)

The CPU can trigger events:

- By using the event lines shared with STREAM and TRGITF, through the CPU shared event generator (CSEG) block.
- By using the dedicated event line input 27 through the CPU-dedicated event generator (CDEG) block.

Similarly, the SPE can trigger events either by using the HSEG, or by using the dedicated event line input 0 through the HDEG block.

Figure 172. SEG block diagram



Generating events with CSEG

The CSEG allows the CPU to reuse events not allocated to a peripheral interface, such as STREAM or TRGITF.

For event inputs 1 to 22, the CPU can select the CSEG as the event generator. This is done through the EVTxSRC field for each event input x.

Then, with a simple write access request to the *HSP CPU shared event generator register (HSP_CSEGR)*, the CPU can trigger one or several events.

The priority encoder of the HSP handles all the activated events according to their priority. The software can check which event is processed by reading back HSP_CSEGR.

Prior to generating an event, the application must enable the CSEG interface by setting CSEGEN in the *HSP BUFF configuration register (HSP_BUFFCFGR)*. When CSEGEN is cleared, the CSEG block is reset and all pending events (**cseg_evt**[x]) are cleared.

Note: There is no protection against overrun situations; the application must ensure that CSEVTx is cleared before setting it again.

For example, if the application wants to activate the event N, it must follow this sequence:

1. Wait for CSEVT[N] = 0. If CSEVT[N] = 1, it means that the HSP did not yet consider the previous request.
2. When CSEVT[N] = 0, set it.

The CSEG block can also be used by the application for debugging purposes, by forcing the execution of a task connected to STREAM or TRGITF. It also allows the CPU to generate events with a high priority level if necessary.

Generating events with CDEG

The CDEG allows the application to trigger any of the 38 tasks through the **cdeg_evt** event.

The CDEG allows one event to be posted at a time: the CPU needs to wait until the previous event has been taken into account before activating a new event.

*Note: The **cdeg_evt** event is routed to the input number 27 of the priority encoder (PE) having the lowest priority.*

Prior to generating events, the application must enable the CDEG interface by setting CDEGEN in the *HSP BUFF configuration register (HSP_BUFFCFGR)*. When CDEGEN is set to 0, the DSEG block is reset, and **cdeg_evt** is cleared.

To generate a dedicated event, the CPU must do the following:

1. Wait until the CDEGBSY bit is cleared.
2. Write the task number to execute (0 to 38) in CTSKN[5:0].

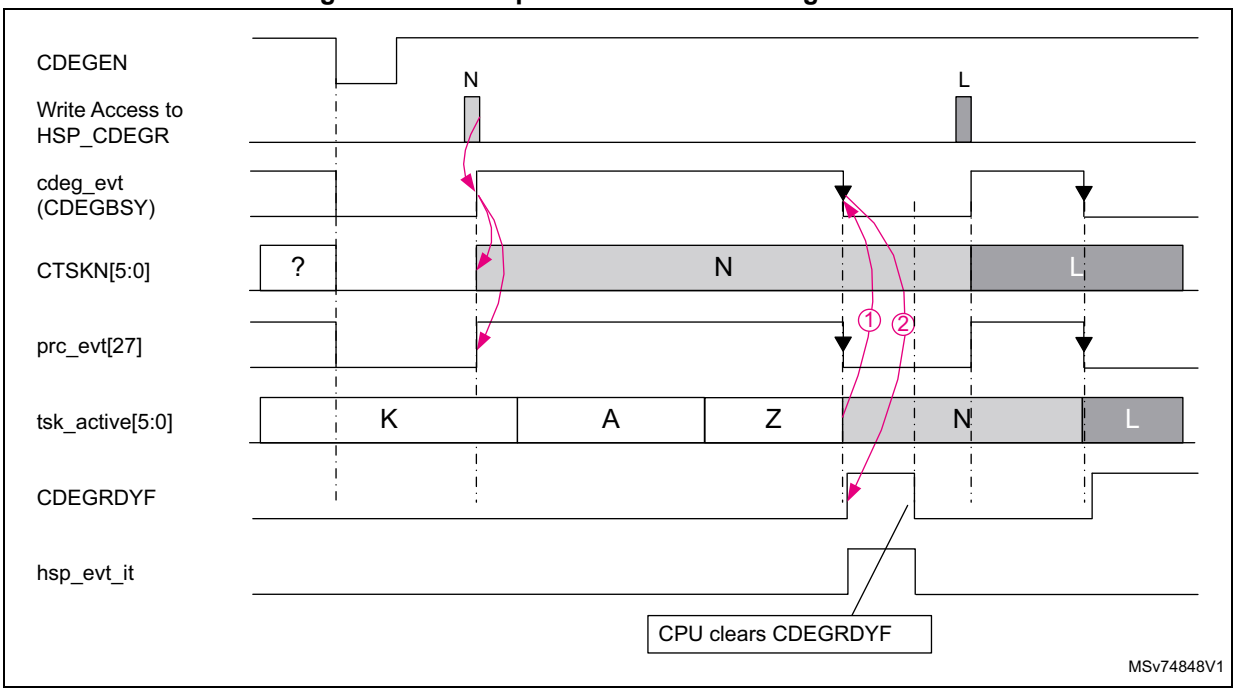
When the CPU writes in the *HSP CPU dedicated event generation register (HSP_CDEGR)*, the **prc_evt**[27] event is asserted, and CDEGBSY is set. The CDEG block can also generate an interrupt when the current CTSKN[5:0] is taken into account by the HSP.

Figure 173 shows a basic example. The CPU writes in HSP_CDEGR and CTSKN[5:0] is set to N. The write operation in this register sets the **cdeg_evt** signal and CDEGBSY flag. **cdeg_evt** is routed to **prc_evt**[27]. When the task controller is able to serve the task associated with **prc_evt**[27], the **cdeg_evt** signal and CDEGBSY flag are cleared by the TCTL (#1 in the figure), and the CDEGRDYF flag is set, asserting an HSP interrupt if the corresponding interrupt enable control bit is set (#2 in the figure). The CDEG block is ready to accept a new request.

*Note: CDEGRDYF is set when a transition from 1 to 0 is detected on the **cdeg_evt** signal.*

CTSKN[5:0] and CDEGBSY are located in the *HSP CPU dedicated event generation register (HSP_CDEGR)*, and CDEGRDYF is located in the *HSP events interrupt status register (HSP_EVT_ISR)*.

Figure 173. Example of dedicated event generation



Generating events with HSEG

The HSP shared event generator (HSEG) allows the SPE to reuse events not allocated to a peripheral interface, such as STREAM or TRGITF.

The HSEG block is very similar to the CSEG, but it is dedicated to SPE usage.

Generating events with HDEG

The HSP dedicated event generator (HDEG) allows the SPE firmware to trigger any of the 38 tasks using the event 0.

The HDEG block is similar to the CDEG, but it is dedicated to SPE usage.

20.4.11 Event controller (EVTC)

The event controller (EVTC) collects the events generated by the various HSP internal peripherals and reconditions them. The event with the highest priority activates the corresponding task, which executes the recorded processing list.

- Handle the STREAM interface.
- Handle the trigger inputs.
- Collect application events.
- Select the event having the highest priority.
- Ask the SPE to execute the processing list associated with the selected event.

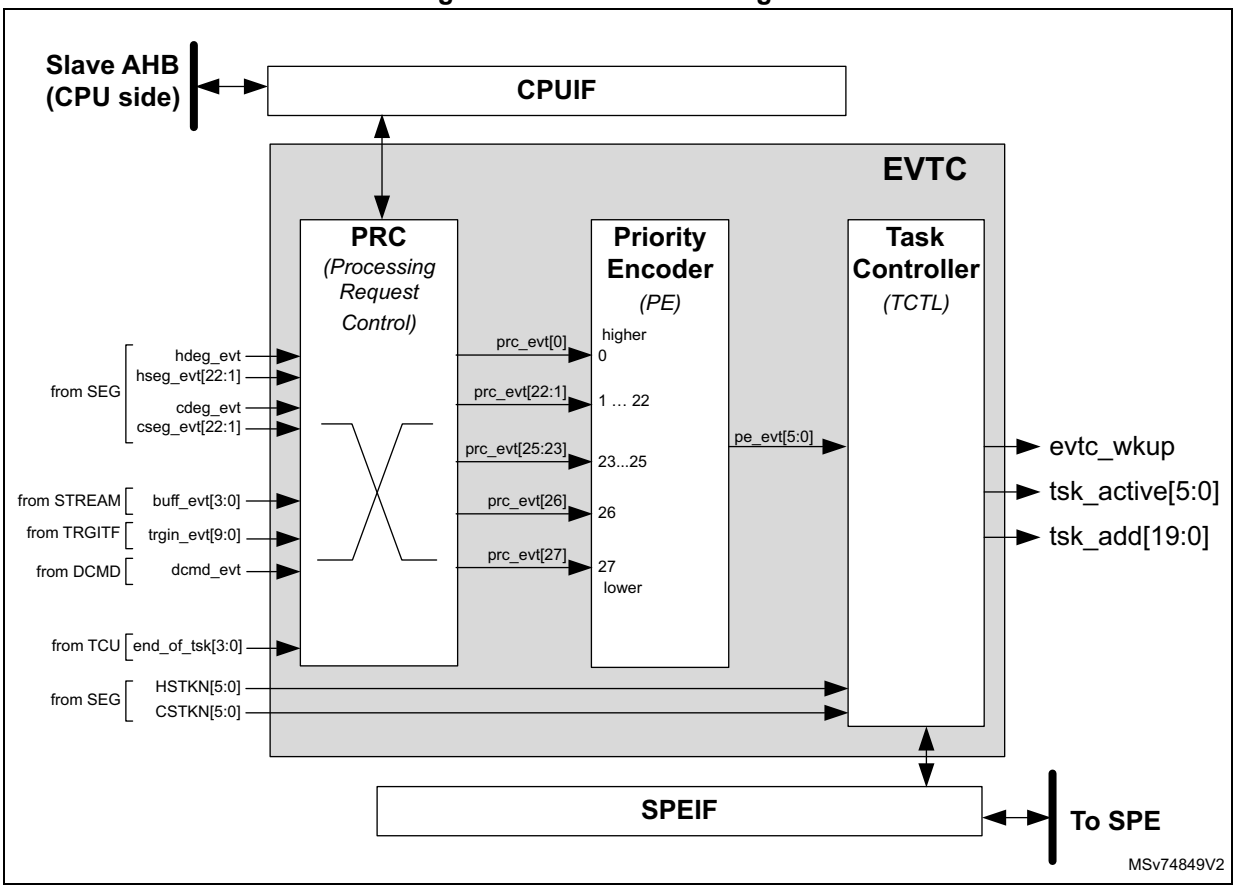
The execution of a processing list can be triggered by several sources:

- The CPU: up to 38 tasks can be triggered by software.
- The SPE: up to 38 tasks can be triggered by firmware.

- Trigger input lines (**hsp_trgi**[X:0]):
  - Triggers from various device peripherals, such as timers, ADCs, DACs, and ADF/MDF
  - Up to 20 trigger inputs are available.
- The STREAM interface:
  - Allows any peripheral to trigger a processing list.
  - Has up to four buffers are available.

The EVTC is configured mainly by the application CPU through the AHB slave interface.

Figure 174. EVTC block diagram



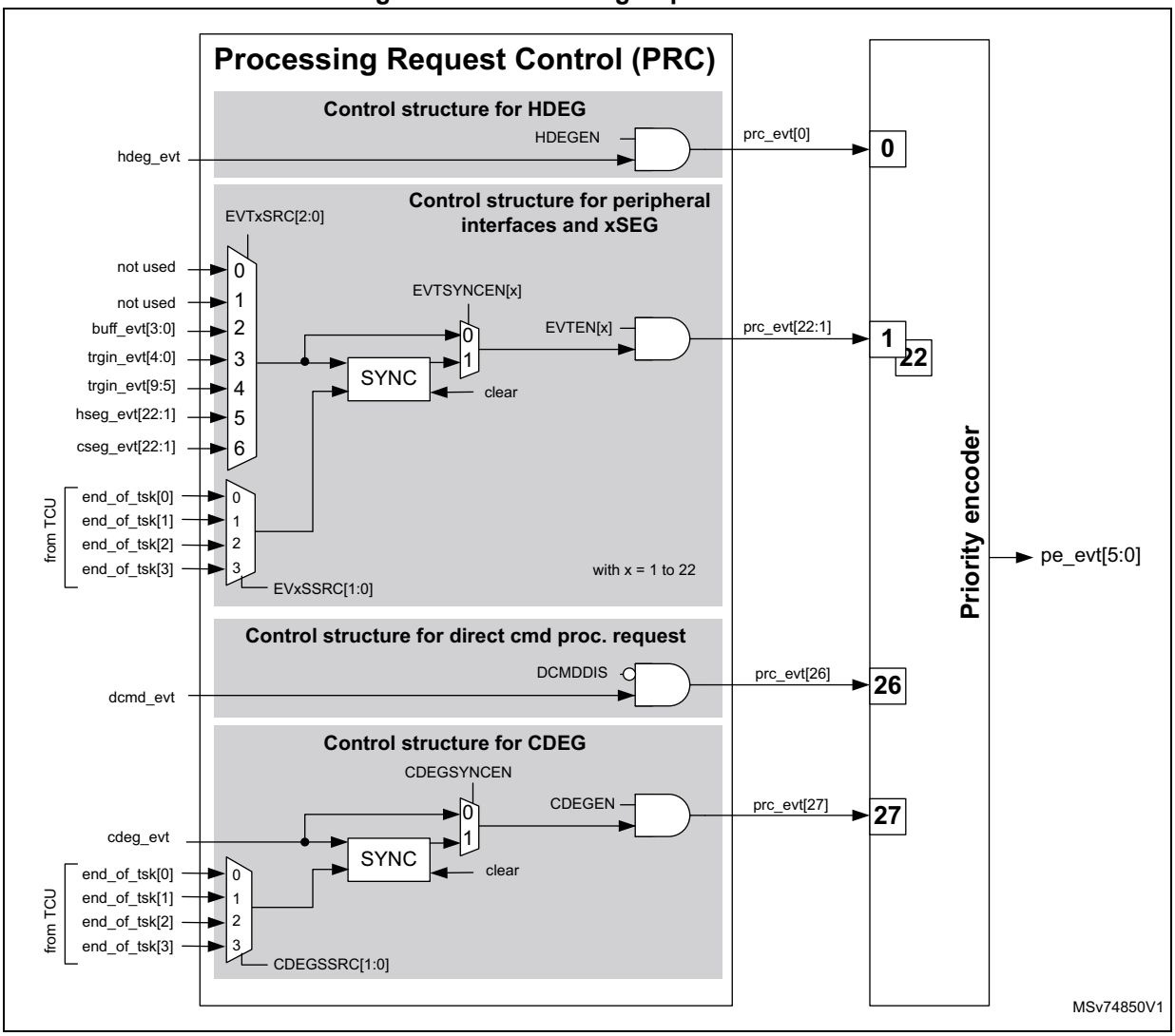
Processing request control (PRC)

For each input of the priority encoder, the processing request control (PRC) is in charge:

- Of selecting the request source that triggers a task through the EVTSRC[2:0]R registers.
- Of enabling and disabling the synchronization option with the *HSP event synchronization enable register (HSP_ESYNCENR)*.
- Of enabling and disabling the generation of the processing request with the *HSP event enable register (HSP_EVTENR)*.

Figure 175 shows the processing request control details for each type of inputs. Most of the inputs have a processing request source selector, a synchronizer, and a processing request enable.

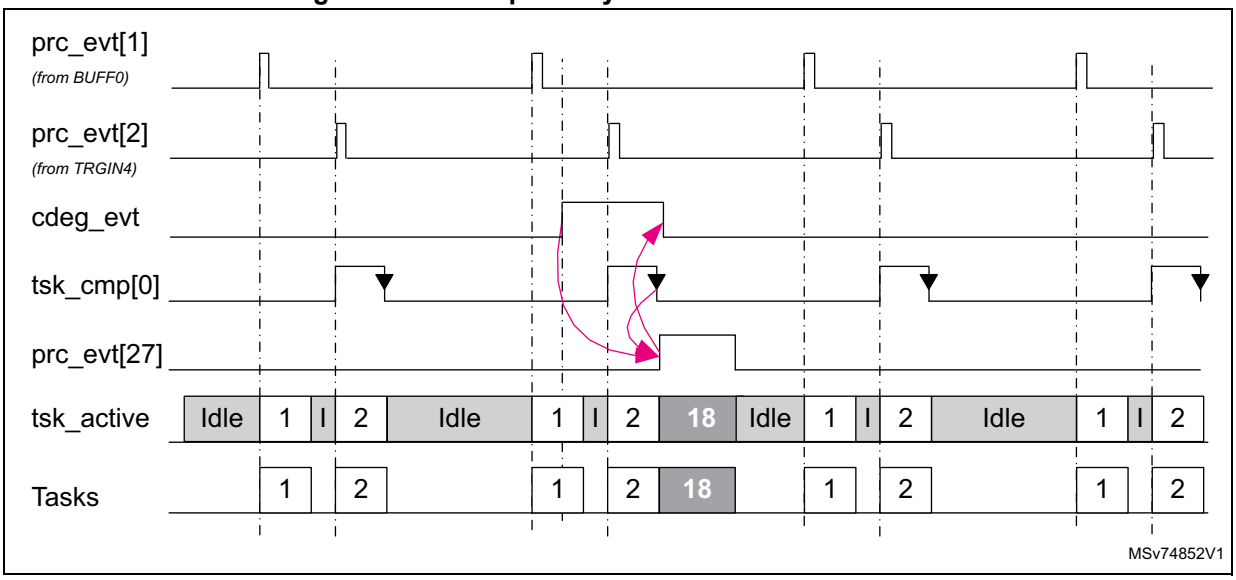
Figure 175. Processing request control



The synchronization mechanism allows the application to delay the activation of an event until the completion of a specific task.

Figure 176 shows an example of the synchronization usage. Tasks 1 and 2 are running at a regular time interval, and their execution must not be delayed. From time to time, the application needs to run an additional task (18 in the figure), for example, to update dynamic parameters. The application does not want to delay the execution of tasks 1 and 2 due to task 18 execution, so it decides to run task 18 right after task 2, because the time interval between the end of task 2 and the start of task 1 allows it. The application sets TSKCMP0[5:0] to 2 to resynchronize the pending **cdeg_evt** event with the end of the execution of task 2.

Figure 176. Example of synchronization mechanism



The table below shows which event can be connected for each priority encoder input, and thus which processing task number is activated (**tsk_active**[5:0]).

Table 143. Trigger sources versus priority and task number(1)

EVTxSRC[2:0] position(2)							Priority or prc_evt[x]	-	Processing task activated
0	1	2	3	4	5	6			
HDEG							0	→	0 to 37
-	-	BUFF[0]	TRGIN[3]	TRGIN[8]	HSEG[1]	CSEG[1]	1	→	1
-	-	BUFF[1]	TRGIN[4]	TRGIN[9]	HSEG[2]	CSEG[2]	2	→	2
-	-	BUFF[2]	TRGIN[0]	TRGIN[5]	HSEG[3]	CSEG[3]	3	→	3
-	-	BUFF[3]	TRGIN[1]	TRGIN[6]	HSEG[4]	CSEG[4]	4	→	4
-	-	BUFF[0]	TRGIN[2]	TRGIN[7]	HSEG[5]	CSEG[5]	5	→	5
-	-	BUFF[1]	TRGIN[3]	TRGIN[8]	HSEG[6]	CSEG[6]	6	→	6
-	-	BUFF[2]	TRGIN[4]	TRGIN[9]	HSEG[7]	CSEG[7]	7	→	7
-	-	BUFF[3]	TRGIN[0]	TRGIN[5]	HSEG[8]	CSEG[8]	8	→	8
-	-	BUFF[0]	TRGIN[1]	TRGIN[6]	HSEG[9]	CSEG[9]	9	→	9
-	-	BUFF[1]	TRGIN[2]	TRGIN[7]	HSEG[10]	CSEG[10]	10	→	10
-	-	BUFF[2]	TRGIN[3]	TRGIN[8]	HSEG[11]	CSEG[11]	11	→	11
-	-	BUFF[3]	TRGIN[4]	TRGIN[9]	HSEG[12]	CSEG[12]	12	→	12
-	-	BUFF[0]	TRGIN[0]	TRGIN[5]	HSEG[13]	CSEG[13]	13	→	13
-	-	BUFF[1]	TRGIN[1]	TRGIN[6]	HSEG[14]	CSEG[14]	14	→	14
-	-	BUFF[2]	TRGIN[2]	TRGIN[7]	HSEG[15]	CSEG[15]	15	→	15
-	-	BUFF[3]	TRGIN[3]	TRGIN[8]	HSEG[16]	CSEG[16]	16	→	16
-	-	BUFF[0]	TRGIN[4]	TRGIN[9]	HSEG[17]	CSEG[17]	17	→	17

Table 143. Trigger sources versus priority and task number(1) (continued)

EVTxSRC[2:0] position(2)							Priority or prc_evt[x]	-	Processing task activated
0	1	2	3	4	5	6			
-	-	BUFF[1]	TRGIN[0]	TRGIN[5]	HSEG[18]	CSEG[18]	18	→	18
-	-	BUFF[2]	TRGIN[1]	TRGIN[6]	HSEG[19]	CSEG[19]	19	→	19
-	-	BUFF[3]	TRGIN[2]	TRGIN[7]	HSEG[20]	CSEG[20]	20	→	20
-	-	BUFF[0]	TRGIN[3]	TRGIN[8]	HSEG[21]	CSEG[21]	21	→	21
-	-	BUFF[1]	TRGIN[4]	TRGIN[9]	HSEG[22]	CSEG[22]	22	→	22
not used							23	→	23
							24	→	24
							25	→	25
DCMD (Direct Command)							26	→	38
CDEG							27	→	0 to 37

1. BUFF[y] represents the events **buff_evt**[y],
   BUFF[y] represents the events **buff_evt**[y],
   TRGIN[y] represents the events **trgin_evt**[y],
   HSEG[y] represents the events **hseg_evt**[y],
   CSEG[y] represents the events **cseg_evt**[y],
2. With x = 1 to 22.

The following table shows a different view: for each event source, the possible priority levels are shown.

Table 144. Trigger sources versus priority and task number(1)(2)

Event sources	Possible priority levels	-	Event sources	Possible priority levels
BUFF[0]	1, 5, 9, 13, 17, 21	→	BUFF[1]	2, 6, 10, 14, 18, 22
BUFF[2]	3, 7, 11, 15, 19	→	BUFF[3]	4, 8, 12, 16, 20
TRGIN[0]	3, 8, 13, 18	→	TRGIN[1]	4, 9, 14, 19
TRGIN[2]	5, 10, 15, 20	→	TRGIN[3]	1, 6, 11, 16, 21
TRGIN[4]	2, 7, 12, 17, 22	→	TRGIN[5]	3, 8, 13, 18
TRGIN[6]	4, 9, 14, 19	→	TRGIN[7]	5, 10, 15, 20
TRGIN[8]	1, 6, 11, 16, 21	→	TRGIN[9]	2, 7, 12, 17, 22
HDEG	0	→	HSEG	1 to 22
CDEG	27	→	CSEG	1 to 22
DCMD	26	→	-	-

1. The HDEG priority is fixed at 0 but can run any task number. The CDEG behaves similarly but has a priority fixed at 27.
2. BUFF[y] represents the events **buff_evt**[y]
   TRGIN[y] represents the events **trgin_evt**[y]
   HSEG[y] represents the events **hseg_evt**[y]
   CSEG[y] represents the events **cseg_evt**[y]

Priority encoder (PE) and task controller (TCTL)

The priority encoder defines which event has the highest priority when several events are requesting the execution of processing lists.

The event priority is fixed. The event having the highest priority is the event connected to input 0 of the PE (dedicated to the SPE). The event having the lowest priority is the event connected to input 27.

When no event is pending, **pe_evt**[5:0], and **tsk_active**[5:0] are equal to 0x3F, and the SPE remains in “wait for event” (WFE).

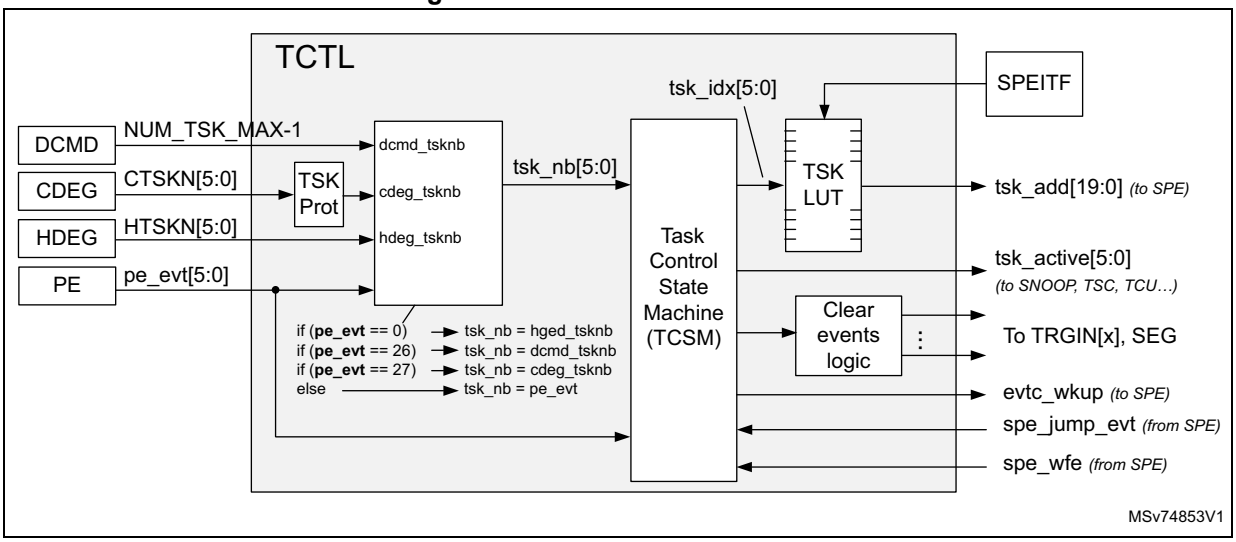
The TCTL generates the task number to be executed (**tsk_nb**[5:0]), according to the output provided by the priority encoder (**pe_evt**[5:0] signal):

- If the output of the priority encoder corresponds to the event number 0 (**pe_evt**[5:0] = 0), the TCTL activates the task number given by HTSKN[5:0]. This field is updated by the SPE when the SPE requests the execution of a specific processing list through the HDEG interface.
- If the output of the priority encoder corresponds to the event number 27 (**pe_evt**[5:0] = 27), then the TCTL activates the task number given by CTSKN[5:0]. This field is provided by the CPU when it requested the execution of a specific processing list through the CDEG interface. Note that a specific protection mechanism prevents having a task number higher than 39. If CTSKN is higher than 37, the task number is forced to 39.
- If the output of the priority encoder corresponds to the event number 26 (**pe_evt**[5:0] = 26), the TCTL activates task number 38. This task is reserved for direct command handling.
- Finally, if the output of the priority encoder provides other values, the TCTL consider it as a task number.

When **tsk_nb**[5:0] becomes different from 0x3F, the TCSM block latches the value of the task to be executed (**tsk_active**[5:0]). The active task value is used as an index to read a look-up table (TSKLUT), which contains the address of the task (or processing list) to be executed by the SPE. The TSKLUT is updated by the SPE. Finally, the TCTL asserts the **evtc_wkup** signal to inform the SPE that a task needs to be executed.

The TCTL is also responsible for clearing TRGINx, SEG, and DCMD events when they are taken into account. When the SPE starts the execution of the selected task, **spe_wfe** goes to zero, and several cycles later the SPE generates a pulse (on **spe_jump_evt**), indicating that the processing list address has been taken into account, and can be updated for the next task to process. In addition, the TCTL generates an error if the SPE does not activate **spe_jump_evt** in the expected time window.

Figure 177. Task controller details



The task having the highest number (task number 39) is dedicated to the SPE firmware and indicates that an unexpected task number has been activated (PLFAIL). The SPE firmware activates the FWERRF flag with the error code described in *Table 145: Firmware error number*. An interrupt is generated to the application if enabled.

When no event is pending, the first event to become active is selected. If several events are waiting for a task completion by the SPE, the event having the highest priority is selected when the SPE acknowledges completion of the previous task.

In the example below, initially there are no pending events and the SPE is in WFE. **tsk_active**[5:0] has a default value output of 0x3F. The **evtc_wkup** signal is inactive, and the output of **tsk_add**[19:0] is the address of the maximum task number supported by the product (see *Section 20.3: HSP implementation*).

As no event is pending, the first active event is the first served. As soon as **prc_evt**[12] becomes active (**trgin_evt**[4]), the priority encoder detects that an event is pending, and immediately updates its output **pe_evt**[5:0] (#1 in the figure). The PESM sees that **pe_evt**[5:0] is no longer equal to the default value, latches **pe_evt**[5:0], asserts **evtc_wkup,** and clears the selected event source if the event is generated by TRGINx, CDEG, HDEG, CSEG, or HSEG.

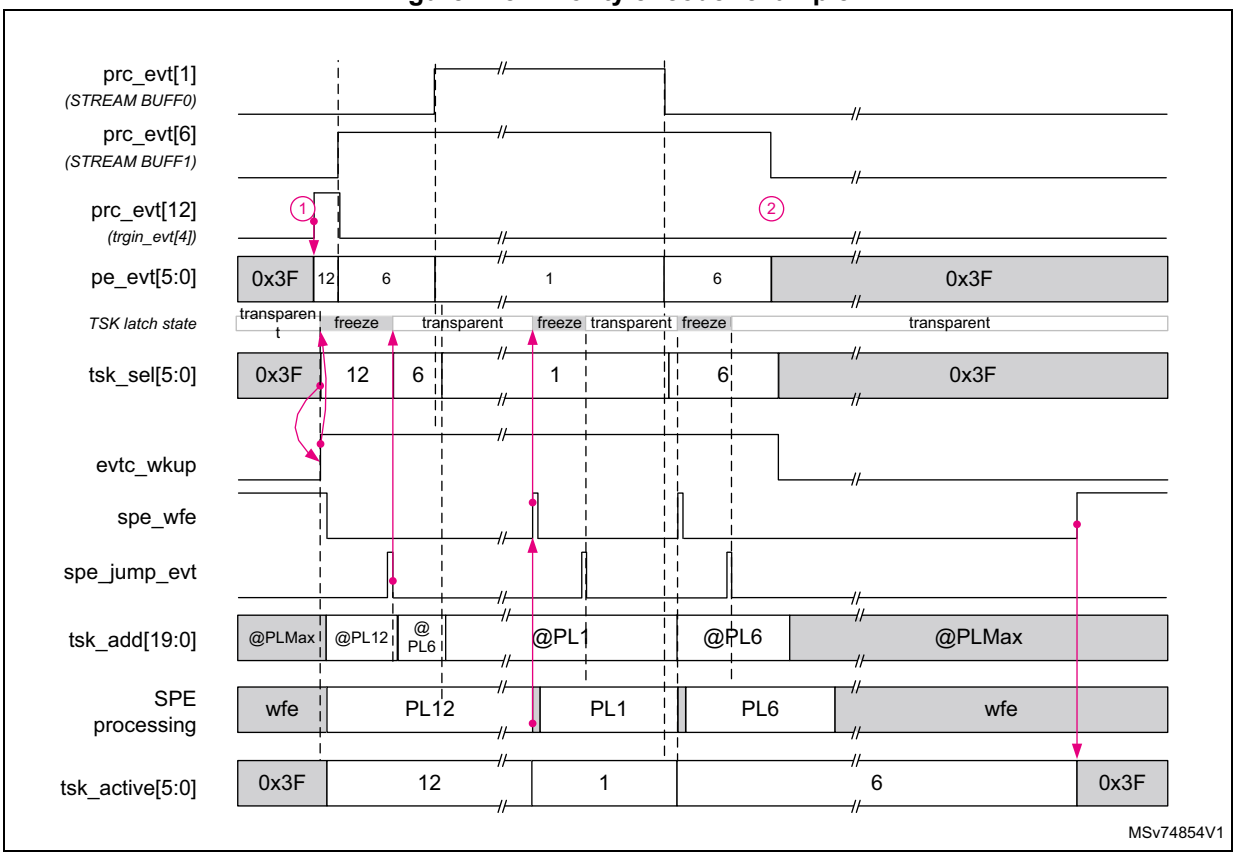
The **pe_evt**[5:0] output is continuously changing according to the event activity.

The SPE exits from WFE (indicated by the clearing of the **spe_wfe** signal) and executes the processing list given at the address **tsk_add**[19:0] (@PL12 in the figure below). The SPE informs the EVTC that **tsk_add**[19:0] can be unfrozen by sending a pulse on **spe_jump_evt** when it branches to the address defined by **tsk_add**[19:0]. The EVTC can select the next processing list to execute while PL12 is under execution. When the execution of PL12 is complete, the SPE attempts to go to WFE, and the signal **spe_wfe** goes to 1 for one cycle. The rising edge of **spe_wfe** indicates to the ETVC that **tsk_add**[19:0] must be frozen again. The PESM latches **pe_evt**[5:0] and keeps **evtc_wkup** activated. In the example, BUFF0 (**prc_evt**[1]) is selected because at the end of the processing of PL12, **prc_evt**[1] has the highest priority.

Seeing **evtc_wkup** still active, the SPE processes PL1. When the SPE completes the execution of the PL1 processing list, it can branch to the execution of PL6.

The **prc_evt**[1] and **prc_evt**[6] are cleared when the SPE reads the corresponding buffer. When the SPE acknowledges **prc_evt**[6] (#2 in the figure), there are no processing requests. The PESM clears **evtc_wkup**. SPE completes the processing of PL6 and goes to WFE.

Figure 178. Priority encoder example



The EVTC monitors the delay between the moment the SPE exits from WFE and the moment it jumps to the execution of the corresponding task. If there is no jump to the selected processing list when the SPE exits from WFE, the EVTC remains frozen. The application is notified of this abnormal behavior by the assertion of the SCHERRF flag. To recover from this situation, the HSP must be reset through the RCC.

The execution of a task cannot be interrupted by another task.

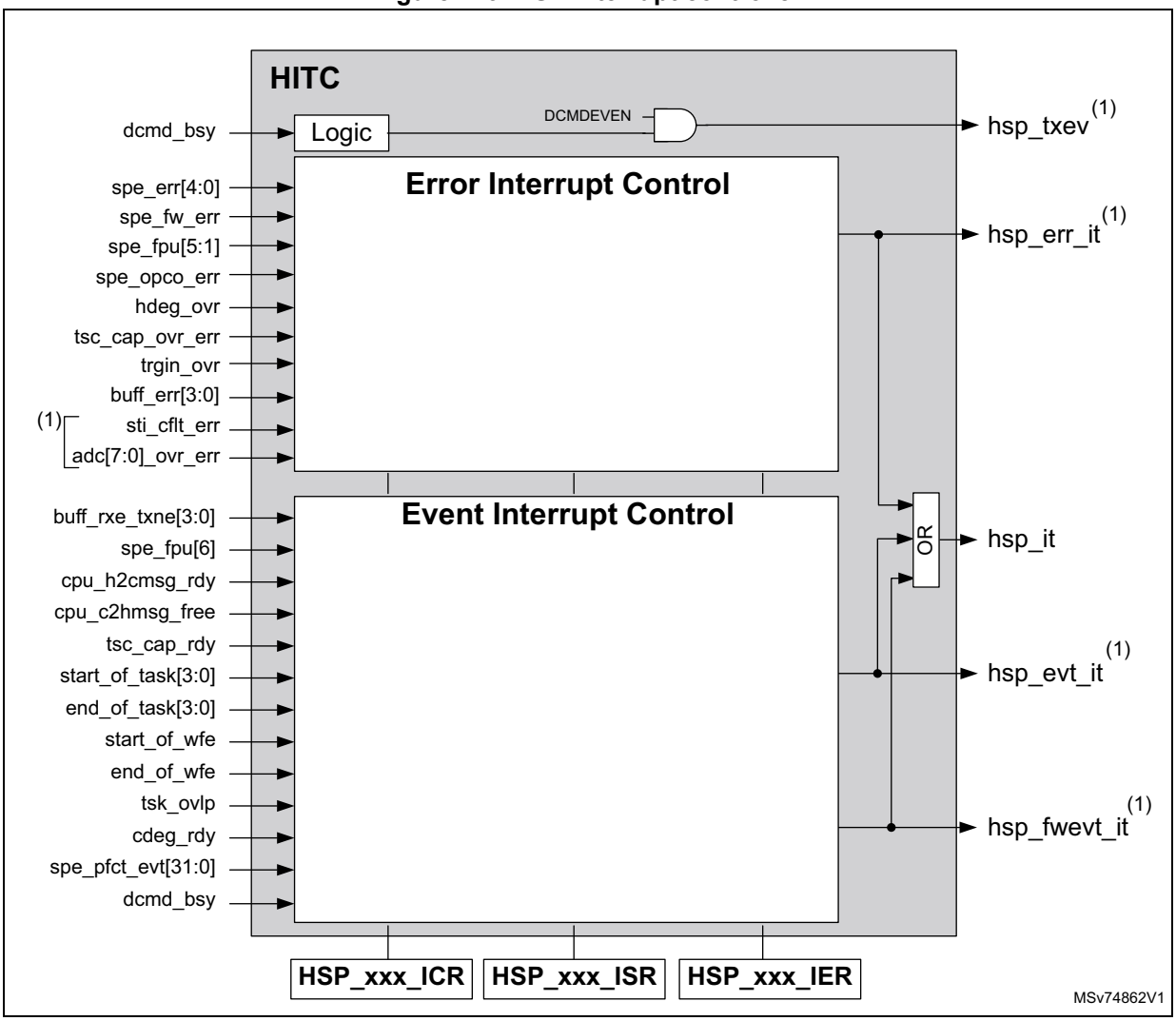
If an enabled event occurs while a task is in execution, the event remains pending until the corresponding task is executed by the SPE.

If several enabled events occur at the same time, the priority encoder defines which must be served first.

20.4.12 HSP interrupt controller (HITC)

The HSP interrupt controller collects events and flags generated by the various parts of the HSP, and generates interrupts to the application processor and to the SPE.

Figure 179. HSP interrupt controller



1. The signals and blocks related to these signals are not always present (see *Section 20.3: HSP implementation*).

Four interrupt lines are provided to the application processor:

- An interrupt line handling errors (**hsp_err_it**)
- An interrupt line handling various hardware events (**hsp_evt_it**)
- An interrupt line handling events generated by the SPE processing (**hsp_fwevt_it**)
- A global interrupt is asserted when one of the previous interrupts is asserted (**hsp_it**)

In addition, the HITC provides a dedicated event output (hsp_txev) able to exit the CPU from a WFE state when a direct command execution is completed. There is no need to acknowledge this event. It can be enabled with DCMDEVEN in the *HSP control register (HSP_CR)*.

The HITC contains several flags, which can be read and cleared at any time by the application. These flags can also generate an interrupt if the corresponding interrupt enables have been set.

The application can enable the interrupts through the *HSP error interrupt enable register (HSP_ERR_IER)*, *HSP events interrupt enable register (HSP_EVT_IER)*, and *HSP processing event interrupt enable register (HSP_PFCTEVT_IER)*.

The application can check the status flags through the *HSP error interrupt status register (HSP_ERR_ISR)*, *HSP events interrupt status register (HSP_EVT_ISR)*, and *HSP processing event interrupt status register (HSP_PFCTEVT_ISR)*.

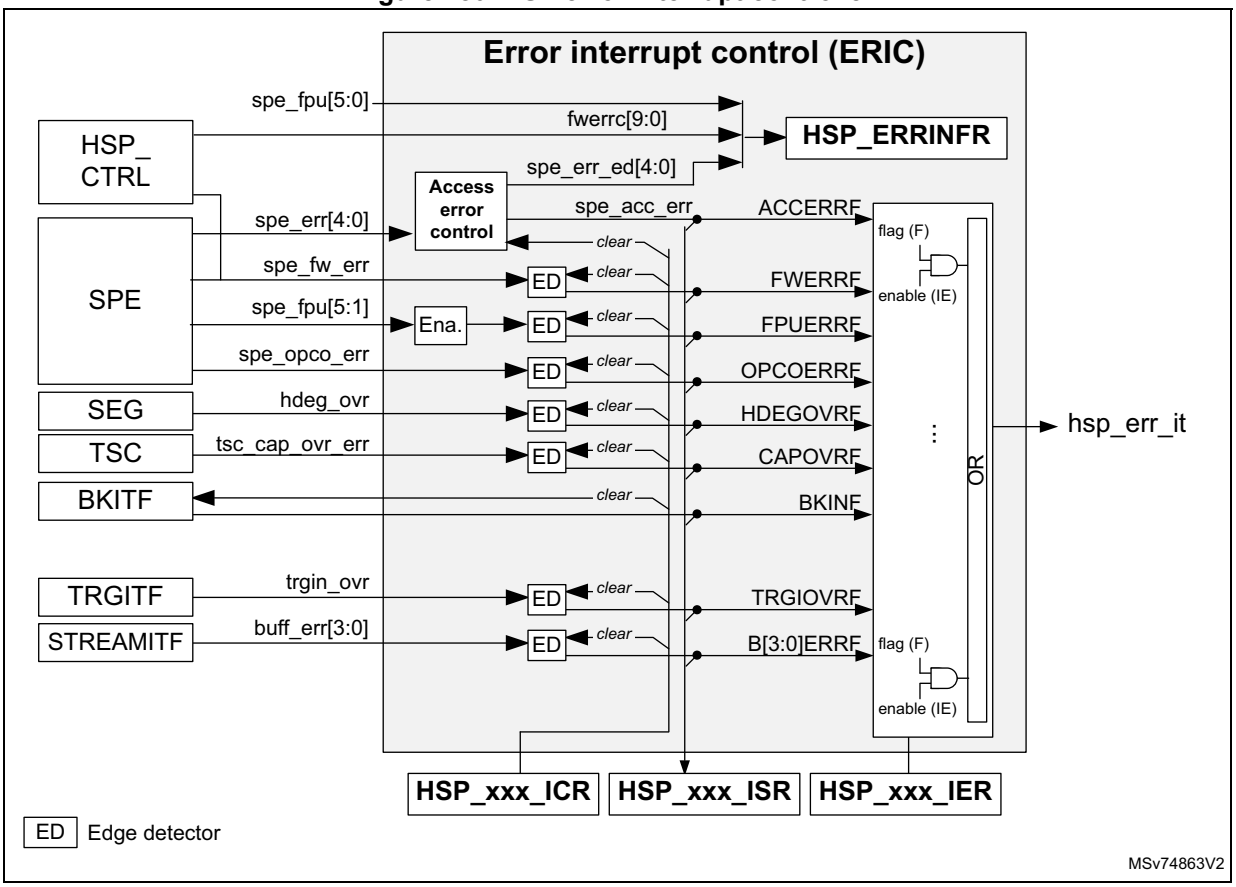
The application can clear the interrupt sources through the *HSP error interrupt clear register (HSP_ERR_ICR)*, *HSP events interrupt clear register (HSP_EVT_ICR)*, and *HSP processing event interrupt clear register (HSP_PFCTEVT_ICR)*.

For additional information on the interrupt root cause, refer to the *HSP error information register (HSP_ERRINFR)*.

Description of error flags

This section lists all event sources capable of asserting the error and global interrupt lines.

Figure 180. HSP error interrupt controller.



Floating point unit error (FPUERRF)

The FPUERRF flag reports floating-point exceptions. This flag is activated when one of the following events occurs: UDFLOW, OVFLOW, DIVZERO, INVALID, or DENORM.

In addition, the *HSP error information register (HSP_ERRINFR)* shows the value of these five exceptions at any time, plus the INEXACT exception.

The six FPU exception signals are generated by the SPE. The meaning of the exceptions is the following:

- **INEXACT**: generated when an operation produces a result that does not fit into its intended destination. The result had to be altered by rounding it off to fit. This exception does not set the flag FPUERRF.

  Examples: $1 + 2^{-25}$ is rounded to 1 with the INEXACT flag activated,

  2.0/3.0 provides inexact rounding as well.
- **UDFLOW**: generated when an operation produces a result closer to 0 than what is supported by the float32. This exception sets the flag FPUERRF if UDFIE is set.
- **OVFLOW**: generated when the largest representable finite number is exceeded in magnitude by what would have been the rounded floating-point result with the exponent range unbounded. This exception sets the flag FPUERRF if OVFIE is set. Example: $2^{127} * 4$ yields an overflow.
- **DIVZERO**: generated when a division by zero is performed. This exception sets the flag FPUERRF if DBZIE is set.
- **INVALID**: generated when an operation produces a result outside of the float32 domain (such as the square root of a negative number or 0/0). This exception sets the flag FPUERRF if INVIE is set.
- **DENORM**: generated when a denormal number is detected. When denormal numbers are detected, they are replaced by 0. This exception sets the flag FPUERRF if DENORMIE is set.

The FPUERRF flag and the information bits UDFLOW, OVFLOW, DIVZERO, INVALID, DENORM, and INEXACT are cleared when FPUERRC is set.

Note: It is possible to freeze HSP execution upon assertion of the FPUERRF flag through the BKITF (see Section 20.4.19: Break interface (BKITF)).

Access error (ACCERRF)

The HSP indicates that an access error occurred by setting ACCERRF. Several conditions can lead to an access error and the *HSP error information register (HSP_ERRINFR)* gives additional information on the root cause. The meaning of the flags present in HSP_ERRINFR is described below:

- CMERR: CRAM/CROM error

  When this flag is set, it means that the SPE fetches or performs load and store operations outside the CRAM/CROM memory range.
- DMERR: BRAM/DRAM error:

  When set, it means that the SPE performs load and store operations outside the BRAM/DRAM memory range.

- MAHBERR internal AHB master error:

  When set, it means that the HSP_CTRL is not properly accessed. This error typically occurs if the SPE performs a wrong access operation to HSP_CTRL.
- SAHBERR slave AHB error:

  When set, it means one of the following:
  - The device tried to access an invalid memory space.
  - The device attempted to write in the CRAM/DRAM memory while it was protected by CDRLCK[1:0].
  - The device performed a nonsupported access operation through the slave AHB interface, such as an access operation not aligned on 32-bit data.

    No error is reported if the device reads or writes into a nonexiting register located in the valid address range of the HSP. When an access operation is not valid, the slave AHB interface always responds to the request with data (in this case 0).

    Invalid read access operations return 0. Invalid write access operations are ignored.

    Accessing above BRAMAB_END (see *Figure 160*) also generates a bus error.

Note: It is possible to freeze HSP execution upon assertion of the ACCERRF flag through BKITF (see Section 20.4.19: Break interface (BKITF)).

Firmware errors (FWERRF)

This flag is activated when the SPE firmware code reported error situations. When the SPE meets an error condition, during the code execution, not covered by hardware error flags, the flag FWERRF is activated and FWERRN[9:0] in the *HSP firmware error register (HSP_FWERR)* is updated.

Note: FWERRN[9:0] is only cleared when the HSP is reset. It always contains the last error code reported by the SPE firmware. Every time the firmware updates the error code, FWERRF is activated, generating a new interrupt if FWERRIE is activated. FWERRF is cleared by the application by setting FWERRC.

Various conditions can lead to the activation of the firmware error flag and FWERRN[9:0] reports the error code detected by the SPE firmware (as detailed in the table below).

Table 145. Firmware error number

FWERRN[9:0] (1)	Error code description
0x000	No error reported, default value after reset
0x005 - 0x008	Reserved
0x009	An event requested the execution of an undefined processing list.
0x00A - 0x3FF	Reserved

1. FWERRN[9:0] is in the *HSP firmware error register (HSP_FWERR)*.

Scheduler errors (SCHERRF)

This flag is activated when the EVTC remains frozen due to an unexpected behavior of the SPE. More precisely, when the EVTC requests the SPE to execute a task, if the SPE responds OK but does not confirm the jump to the task address, the EVTC remains frozen. When this situation occurs, the SCHERRF flag is activated. The only way to recover from this situation is to reset the HSP.

Invalid OpCode error (OPCOERRF)

This flag is activated when the SPE detects an invalid opcode. The SPE executes a NOP for each invalid opcode detected.

Note: It is possible to freeze HSP execution upon assertion of the OPCOERRF flag through BKITF (see Section 20.4.19: Break interface (BKITF)).

STREAM error (B[3:0]ERRF)

For buffers configured in the C2H direction, this flag represents an overrun. For buffers configured in the H2C direction, it represents an underrun.

This flag is activated when new data are written in C2HBUFFx by the CPU, while the previous data have not yet been read. In this case, the new data are written in C2HBUFFx.

This flag is activated when H2CBUFFx is read by the CPU, while it is empty (HSP did not write any new data on time). In this case, the CPU reads the previous data.

TRGI overrun error (TRGIOVRF)

This flag is activated when a new event occurs while the previous event is still pending. The application can check which TRGIN interface generated the overrun through TRGIERR[9:0] in the *HSP error information register (HSP_ERRINFR)*.

HSP-dedicated event generator overrun error (HDEGOVRF)

This flag is set when the SPE posts a firmware event while the previous event has not yet been taken into account. The new event posted by the SPE is ignored.

Capture overrun error (CAPOVRF)

This flag is set when a new capture value must be written in HSP_CAPDR while the device has not yet read the previous one. In this case, the new capture value is dropped.

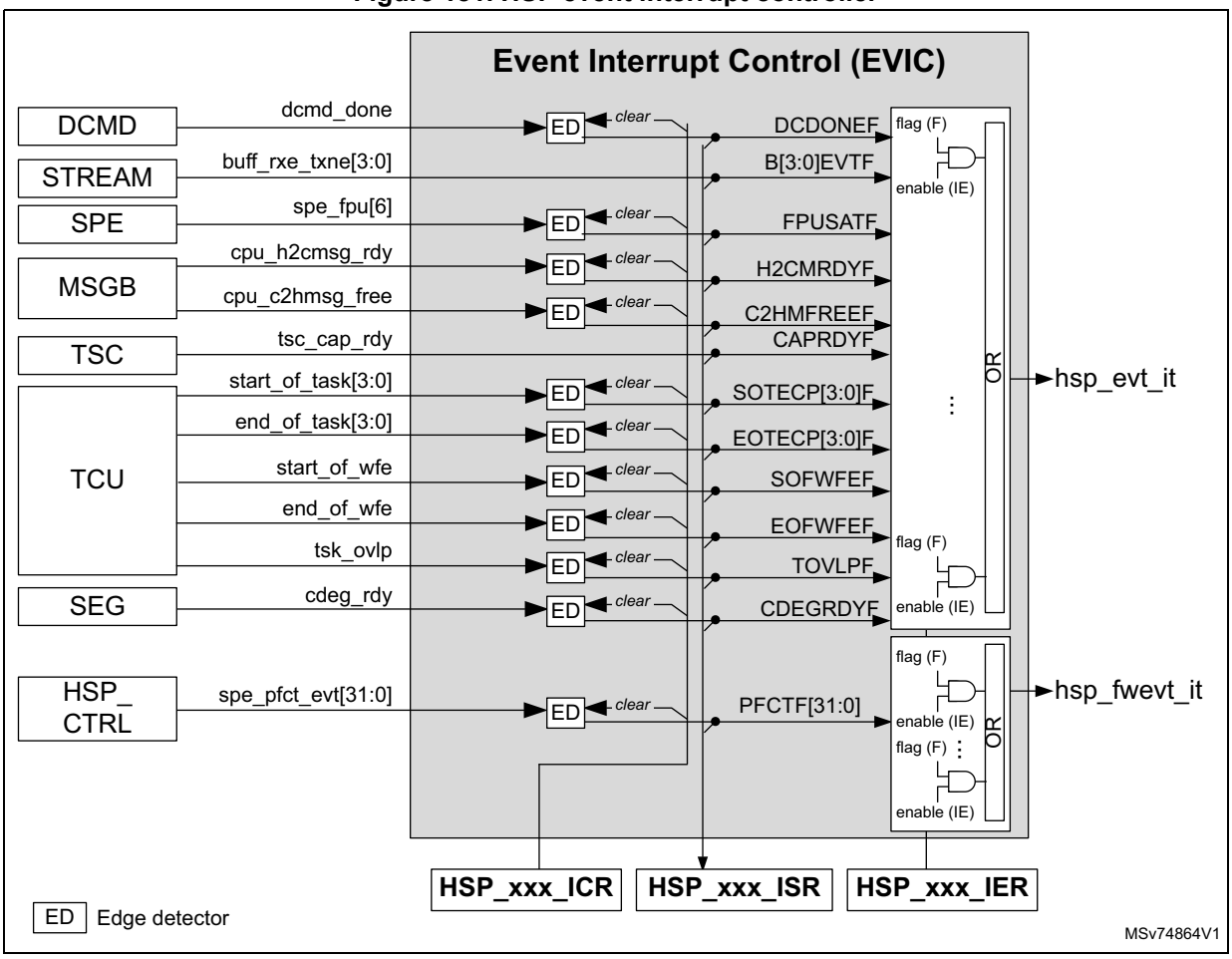
Break input (BKINF)

This flag indicates that the HSP receives a break input event. The SPE is frozen and an application action is needed to restart the HSP (see *Section 20.4.19: Break interface (BKITF)*).

Description of event flags

This section lists all the event sources capable of asserting the event and global interrupt lines.

Figure 181. HSP event interrupt controller



CPU-dedicated event generator ready (CDEGRDYF)

It indicates if the execution of the previous event posted by the CPU into the CDEG block has been started. The CPU can use this information to post a new event.

STREAM buffer event (B[3:0]EVTF)

For buffers configured in the C2H direction, this flag indicates that the buffer is empty. For buffers configured in the H2C direction, this flag indicates that the buffer is not empty.

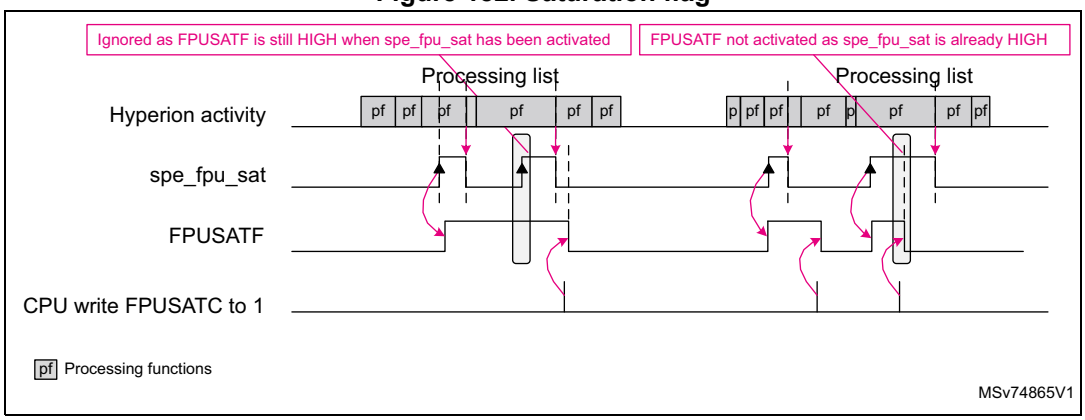
FPU saturation (FPUSATF)

This flag indicates that the FPU detected and performed a saturation of the signal.

As shown in the figure below, when a saturation is detected, the SPE sets the **spe_fpu_sat** signal. This signal remains set until the completion of the current processing function. The

FPUSATF flag is set when a rising edge on **spe_fpu_sat** is detected, and it is cleared when FPCSATC is set.

Figure 182. Saturation flag



Capture buffer ready (CAPRDYF)

It indicates that a new capture value is available in the *HSP data capture register (HSP_CAPDR)*.

Message box ready and free (H2CMRDYF, C2HMFREEF)

The H2CMRDYF flag indicates that a message is available for the CPU in the MSGB.

The C2HMFREEF flag indicates that the MSGB is free and the CPU can post a new message for the HSP.

Start/end of task event comparator (SOTECP[3:0]F, EOTECP[3:0]F)

SOTECPxF is set if the task selected by TSKCMPx is starting.

EOTECPxF is set if the task selected by TSKCMPx has just been completed.

Start/end of WFE state (EOFWFEF, SOFWFEF)

SOFWFEF is set when the SPE enters the WFE state.

EOFWFEF is set when the SPE exits the WFE state.

Task overlap (TOVLPF)

This flag is set when the execution of one of the tasks on the selected range has been delayed due to the execution of another task (see *Section 20.4.14: Task comparator unit (TCU)*).

Direct command done flag (DCDONEF)

This flag is set when the execution of the current direct command is completed. More precisely, it is set when the DCBSY flag transitions from 1 to 0 (see *Section 20.4.7: Direct command interface (DCMD)*).

Processing function flags (PFCTF[31:0])

These flags are activated by the processing functions, several of which use flags.

The table below gives a summary of all interrupts that the application can use.

Table 146. HSP interrupt requests

Interrupt acronym[1]		Interrupt event	Event Flags	Enable control bits	Interrupt clear method	Exit Sleep	Exit Stop and Standby
HSP	HSP_ERR[2]	C2HBUFF[3:0] overrun or H2CBUFF[3:0] underrun	B[3:0]ERRF	B[3:0]ERRIE	Write 1 in B[3:0]ERRC	Yes	No
		HDEG event overflow	HDEGOVRF	HDEGOVRIE	Write 1 in HDEGOVRC	Yes	No
		Access error	ACCERRF	ACCERRIE	Write 1 in ACCERRC	Yes	No
		SPE invalid opcode	OPCOERRF	OPCOERRIE	Write 1 in OPCOERRC	Yes	No
		SPE arithmetic error	FPUERRF	FPUERRIE	Write 1 in FPUERRC	Yes	No
		Capture register overflow	CAPOVRF	CAPOVRIE	Write 1 in CAPOVRC	Yes	No
		SPE firmware errors	FWERRF	FWERRIE	Write 1 in FWERRC	Yes	No
		Scheduler error	SCHERRF	SCHERRIE	Write 1 in SCHERRC	Yes	No
		Break input activated.	BKINF	BKINIE	Write 1 in BKINC	Yes	No
		Task overlap detection	TOVLPF	TOVLPIE	Write 1 in TOVLPC	Yes	No
	HSP_EVT[2]	C2HBUFF[3:0] empty or H2CBUFF[3:0] not empty	B[3:0]EVTF	B[3:0]EVTIE	Write or read HSP_BUFFxDR	Yes	No
		Capture buffer ready flag	CAPRDYF	CAPRDYIE	Read HSP_CAPDR	Yes	No
		Message ready for CPU	H2CMRDYF	H2CMRDYIE	Write 1 in H2CMRDYC	Yes	No
		Message box to HSP free	C2HMFREEF	C2HMFREEIE	Write 1 in C2HMFREEC	Yes	No
		Saturation from FPU	FPUSATF	FPUSATIE	Write 1 in FPUSATC	Yes	No
		Start of task 0 to 3	SOTECP[3:0]F	SOTECP[3:0]IE	Write 1 in SOTECP[3:0]C	Yes	No
		End of task 0 to 3	EOTECP[3:0]F	EOTECP[3:0]IE	Write 1 in EOTECP[3:0]C	Yes	No
		Start of WFE	SOFWFEF	SOTWFEIE	Write 1 in SOTWFEC	Yes	No
		End of WFE	EOFWFEF	EOFWFEIE	Write 1 in EOTWFEC	Yes	No
		Direct command busy	DCDONEF	DCDONEIE	Write 1 in DCDONEC	Yes	No
		CPU dedicated event generator ready	CDEGRDYF	CDEGRDYIE	Write 1 in CDEGRDYC	Yes	No
	HSP_FWEVT[2]	Processing functions flags	PFCTF[31:0]	PFCTIE[31:0]	Write 1 in PFCTC[31:0]	Yes	No

1. HSP_ERR is generated by the **hsp_err_it** signal; HSP_EVT is generated by the **hsp_evt_it** signal; HSP_FWEVT is generated by the **hsp_fwevt_it** signal; HSP_GLOB is generated by the **hsp_it** signal.
2. This interrupt line is not always connected to the interrupt controller (see *Section 20.3: HSP implementation*).

20.4.13 HSP reset and clocking

The HSP MMC block is reset by the product system reset.

The other HSP blocks, including the SPE, are also reset by the product system reset, but can also be reset by the application through a reset control bit in the RCC.

When the application resets the HSP with the RCC control bit, the following conditions must be satisfied:

- There are no ongoing transactions through its AHB slave interface.
- The SPE no longer performs transactions through its bus interfaces.

The HSP offers a mechanism to ensure that the SPEhas stopped all transactions before a reset. The application must ensure that transactions through its AHB slave interface are stopped.

The application must follow this sequence to set the HSP to a safe state before resetting it with the RCC control bit:

1. Clear the HSP_EVTENR register.
2. In the HSP_ITFENR register, set the DCMDDIS bit and clear all others.
3. Set RSTREQ in HSP_CR.
4. Wait for RSTOK = 1 in HSP_CR.

   This bit must go high after 1 ms at most after an RSTREQ assertion. A timeout must control if this delay elapsed.

   If RSTOK is set, the application can reset the HSP safely through the RCC reset control bits, by executing the following sequence:

   a) Disable the HSP clock (**hsp_hclk_core_ck**).

   b) Set the HSP reset bit to 1.

   c) Clear the HSP reset bit.

   d) Enable the HSP clocks.

   If the timeout elapsed before RSTOK is set, it cannot be ensured that resetting the HSP does not disturb the activity of other components of the product.

HSP clocking

The HSP works with two clocks that are synchronous to the interconnect clock:

- **hsp_hclk_mem_ck:** clock for the memory components and associated logic

  This clock allows the application to use the SRAM4 as additional memory even when the HSP is disabled.
- **hsp_hclk_core_ck:** clock for the remaining parts of the HSP, including the SPE

When the HSP is disabled or not used (BOOTEN = 0), the CPU must provide **hsp_hclk_mem_ck** to access the SRAM4.

When the application wants to use the HSP, the CPU must provide **hsp_hclk_mem_ck** and **hsp_hclk_core_ck**.

The clock generation is controlled through the RCC.

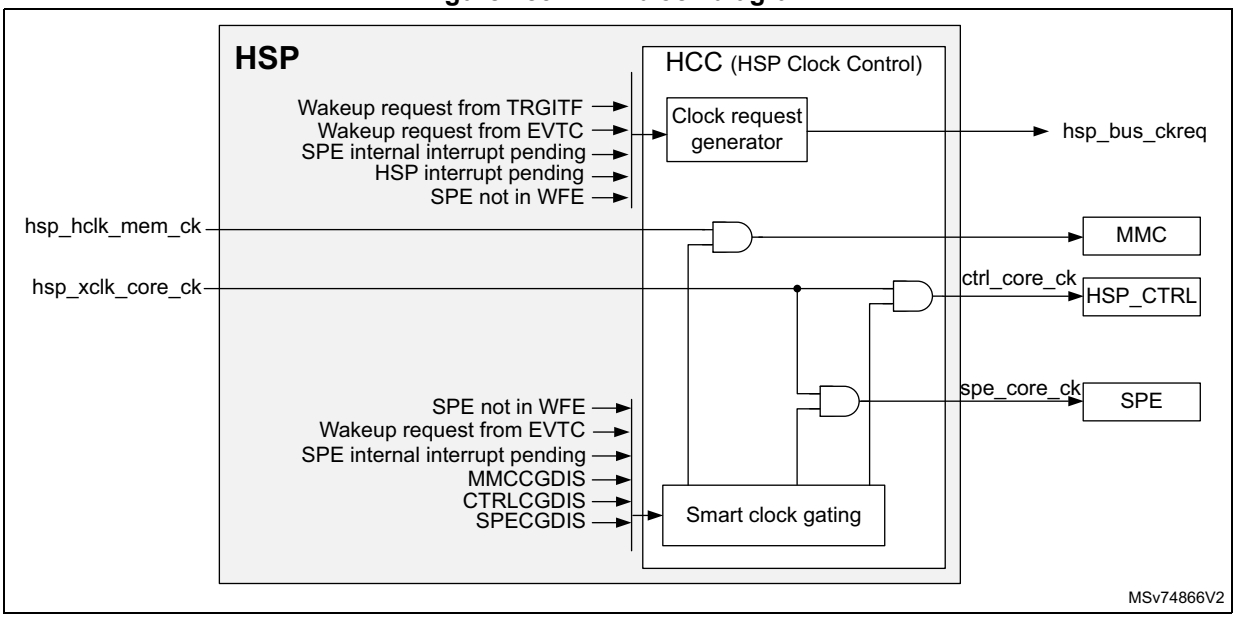
Smart clock gating

The HSP performs a local clock gating to reduce power consumption. This smart clock gating is activated by default, but can be disabled by CTRLCGDIS, SPECGDIS and MMCCGDIS in the *HSP control register (HSP_CR)*.

- When CTRLCGDIS is cleared, parts of the HSP_CTRL block are not clocked when they are inactive.
- When MMCCGDIS is cleared, the MMC block is not clocked when there is no memory access.
- When SPECGDIS is cleared, the SPE is not clocked in WFE.

Setting one of these bits forces the clocking of the corresponding block.

Figure 183. HCC block diagram



The HSP can request the bus clock (with the **hsp_bus_ckreq** signal) when needed.

In addition, the HSP can detect a TRGIN event, even if **hsp_hclk_mem_ck** and **hsp_hclk_core_ck** clocks are not provided by the product.

See the low-power capability table “Functionalities depending on the working mode” in the PWR section of the product to check if the bus clock request is supported.

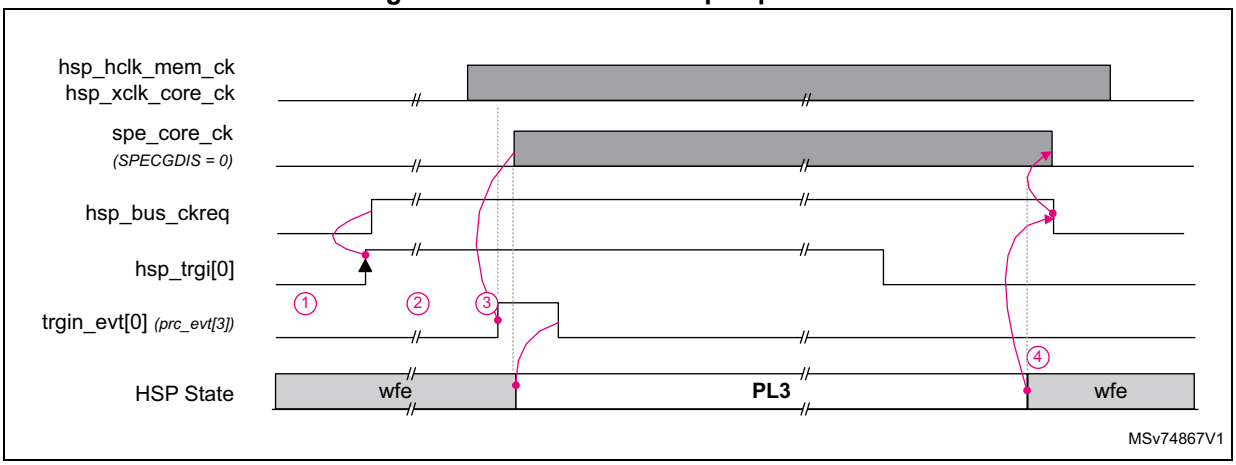
Figure 184 shows a wake-up example using TRGITF.

1. Before the trigger event occurs, the HSP is not clocked at all (#1 in the figure).
2. As soon as the **hsp_trgi**[0] active edge is detected asynchronously, the HSP asserts the **hsp_bus_ckreq** signal. After a setup delay generally due to the startup of the

oscillator providing the clock, the RCC provides a bus clock, but the **spe_core_ck** remains gated (#2 in the figure).

3. Using the bus clock, the EVTC can generate **trgin_evt**[0] and wakes up the SPE. **spe_core_ck** is enabled to allow the SPE to process processing list PL3 (#3 in the figure).
4. **hsp_bus_ckreq** is maintained active until the SPE turns back to WFE. The **spe_core_ck** clock is gated as well (#4 in the figure).

Figure 184. TRGITF wake-up capabilities



20.4.14 Task comparator unit (TCU)

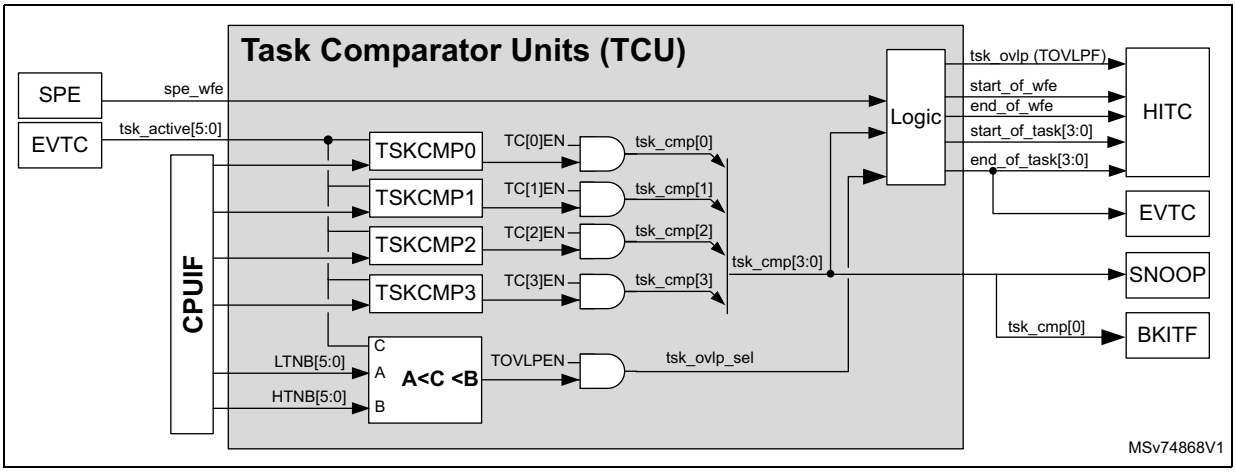
The TCU offers several services based on the task active value (tsk_active[5:0]) provided by the EVTC.

The TSKCMP[3:0] comparator enables the following features controlled through the *HSP task comparator unit register (HSP_TCUCFGR)*:

- Synchronize the generation of an event with the completion of a selected task.
- Generate an interrupt when the execution of a specific task starts or stops, or when the SPE goes to WFE.
- Route comparator outputs to I/Os (with HSP_SNP[5:0] I/Os) for debugging or tuning purposes, to check the duration of a task or for task sequencing.

The TCU can also be used to detect if there are task overlap situations. This function informs the application that the execution of several tasks does not occur at the expected time, because the HSP was performing another task. In some applications, this may inform the application that scheduling behaves in an unexpected way.

Figure 185. TCU block diagram

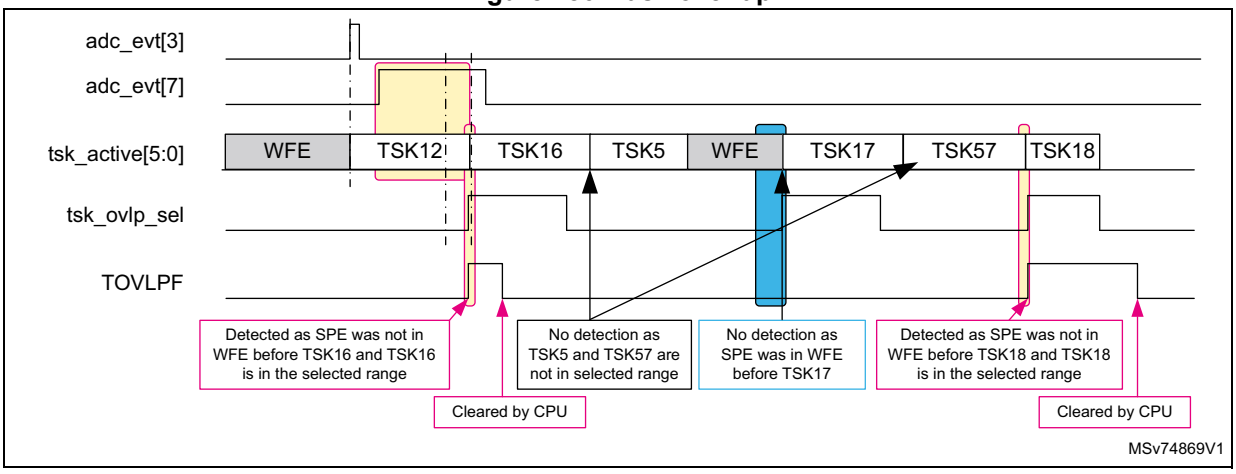


A task overlap situation is detected if between the end of execution of the previous task and the start of the next task, the SPE remains only one clock cycle to WFE. This means that when the previous task execution was completed, there was a request to execute another task. If the delayed task is not in the selected group, the detection is not activated.

The application can define a group of tasks where the overlap situation must be reported; the group range can go from task number 1 to task number 39.

In the figure below, HTNB is set to 19, and LTNB is set to 15, to restrict the test overlap check to tasks 16, 17, and 18. The tasks 16, 5, 57, and 18 have been delayed, but only the delay of tasks 16 and 18 are reported.

Figure 186. Task overlap



Note: In some cases, it is normal to have task overlap situations: for example, when the SPE triggers a new task, or when a task is synchronized with the completion of another task. Selecting a limited group of tasks where the overlap must be checked reduces the risk of having spurious generation of the task overlap event.

To activate the task overlap function, the application must do the following:

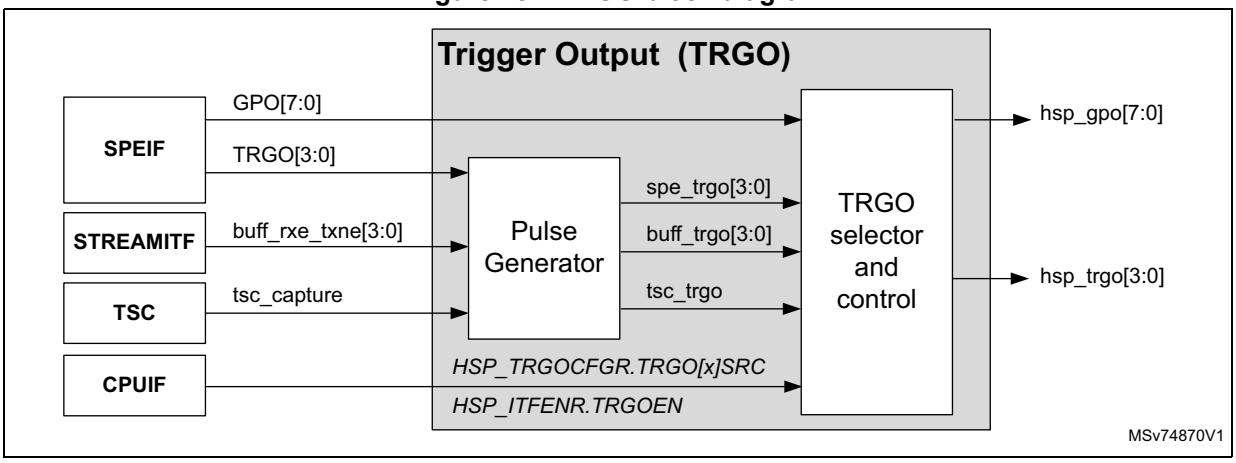
1. Select the range of tasks where the overlap condition must be checked, by initializing HTNB[5:0] and LTNB[5:0] in the *HSP task overlap control register (HSP_TOVLPCR)* to the range boundary.
2. Set the TOVLPIE control bit if an interrupt is requested from the CPU.
3. Set TOVLPEN in the *HSP task overlap control register (HSP_TOVLPCR)*.

20.4.15 Trigger outputs (TRGO)

The HSP offers the possibility to generate trigger pulses on up to four trigger outputs (**hsp_trgo**[3:0]). The pulse width is one **hsp_hclk_core_ck** period.

It is also possible for the application to control the level of eight GPOs (**hsp_gpo**[7:0]) using dedicated processing functions.

Figure 187. TRGO block diagram



The TRGO is enabled with TRGOEN in the *HSP interface enable register (HSP_ITFENR)*. When TRGOEN=0, it is not possible to generate trigger pulses on **hsp_trgo**[3:0], nor to change the level of **hsp_gpo**[7:0].

A selector allows the application to define the trigger source for one of the four trigger outputs, using the *HSP TRGO configuration register (HSP_TRGOCFGR)*.

The table below lists the possible trigger sources.

Table 147. List of trigger sources

Trigger source	Description
TRGO[3:0]	The SPE can generate a trigger pulse by setting one of those 4 bits.
buff_rxe_txne[3:0]	A trigger pulse is generated in the corresponding trigger output (**buff_trgo**[3:0]), every time C2HBUFF[3:0] is ready to accept a new data from the CPU, or every time H2CBUFF[3:0] contains a data for the CPU.
tsc_capture	A trigger pulse generated every time a new capture data is available,

The trigger output can typically be used to trigger an automatic data transfer using a DMA device. For example:

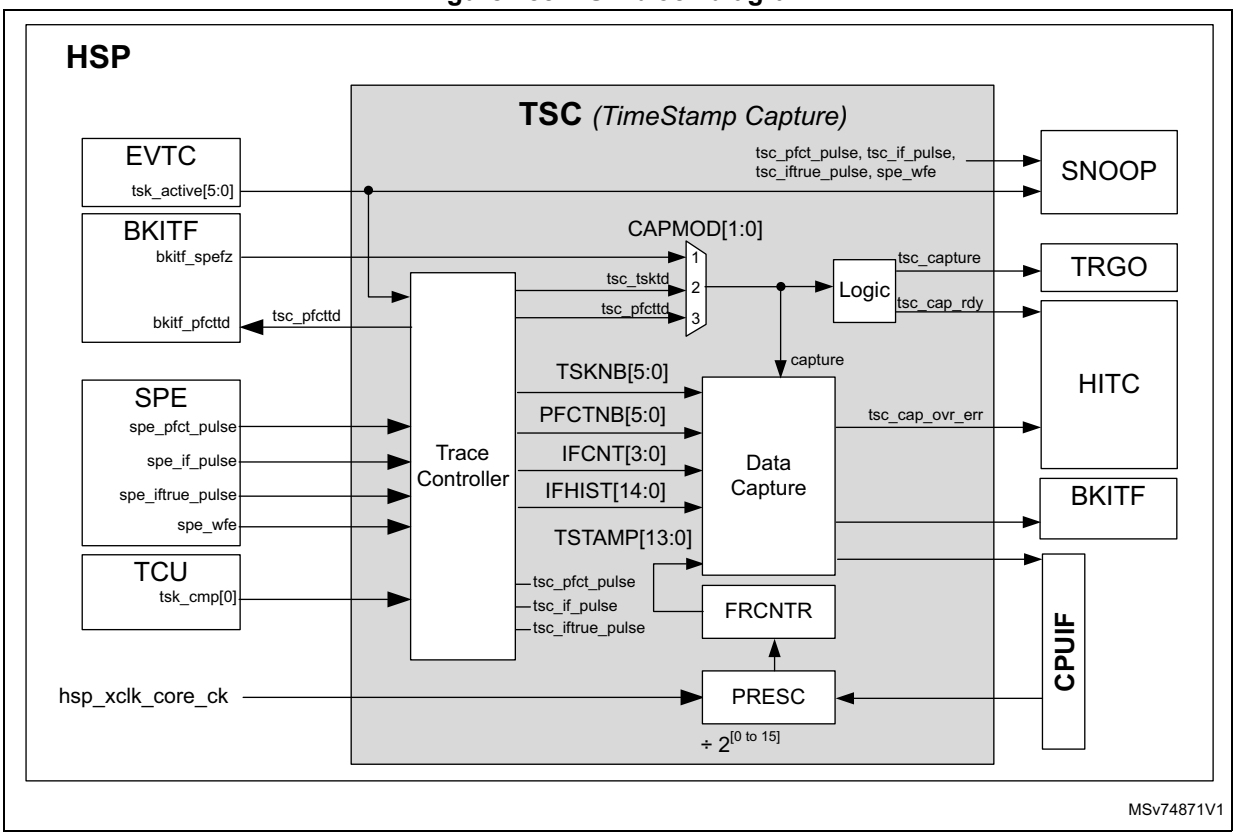
- When a pulse is generated from the **tsc_capture** event, the DMA device can read the *HSP data capture register (HSP_CAPDR)* and write it into a circular buffer, storing the scheduling history automatically.
- If a buffer configured in the C2H direction generates a pulse trigger, the DMA device can automatically write the next data to process in the *HSP BUFFx data register x (HSP_BUFFxDR)*. The SPE is informed that a new sample is available on one of the Rx buffers.
- If a buffer configured in the H2C direction generates a pulse trigger, the DMA device can automatically read the corresponding Tx buffer through the *HSP BUFFx data register x (HSP_BUFFxDR)*. The SPE is informed that a Tx buffer is available and a new sample can be written in the corresponding Tx buffer.

20.4.16 Time-stamp capture (TSC)

To ease the processing tuning and the debug, the HSP offers the possibility to capture a time-stamp, the task number, the processing function number, and other helpful information, using three different capture modes:

- Task capture mode: a capture is done every time the activated task number changes.
- Processing function capture mode: a capture is done every time the processing function number changes.
- Break capture mode: a capture is done every time the SPE breaks.

Figure 188. TSC block diagram



The TSC mainly contains a trace controller, a data capture function, and a free-running counter.

The data capture function consists of storing information concerning SPE activity in the HSP_CAPDR register.

The data capture function can be controlled with CAPMOD[1:0]:

- When CAPMOD[1:0] = 0, the capture function is disabled. The capture events are ignored. The complete capture function is no longer clocked.
- When CAPMOD[1:0] = 1, a capture is performed every time the SPE breaks (**bkitf_spefz**).
- When CAPMOD[1:0] = 2, a capture is performed every time the processing list number changes (**tsk_active**[5:0]).
- When CAPMOD[1:0] = 3, a capture is performed every time the processing function number changes (**tsc_pfctnb**[5:0]).

The SPE provides several signals to ease the debugging:

- spe_pfct_pulse signal generates a pulse when the SPE starts the execution of a processing function.
- spe_if_pulse signal generates a pulse when a conditional function is executed.
- spe_iftrue_pulse signal generates a pulse if the conditional function result is true.
- spe_wfe signal indicates if the SPE is in WFE.

All these signals are visible through the SNOOP block (see *Section 20.4.18: Snoop function (SNOOP)*).

The TSC trace controller reconditions the signals received from the SPE, TCU, and EVTC and provides the following services:

- Counting the processing function inside a processing list (tsc_pfctnb[5:0]). This counter is reset every time the SPE goes back to WFE. When this counter reaches 63, on the next processing function it wraps around to 0 and continues counting.
- Generating a signal indicating when the execution of a new processing function starts. This signal (tsc_pfcttd) can be used to freeze the SPE every time the execution of a new processing function starts.

  It can also be used to generate a capture event and thus monitor the execution of processing functions.

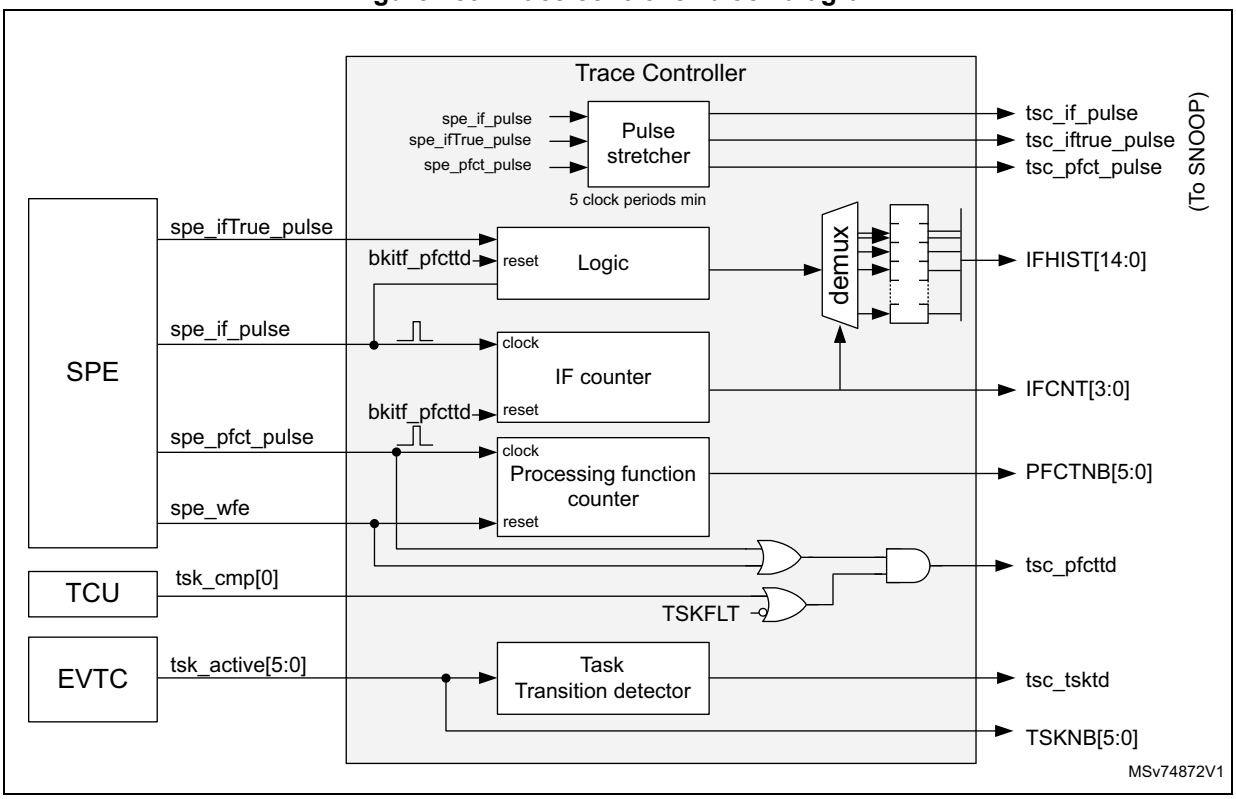
  In addition, it is possible to generate the tsc_pfcttd signal only during the execution of a specific task number. This can be done by using the output of TSKCMP0 (tsk_comp[0]) as the filter signal. This filtering function can be controlled through the TSKFLT bit.
- Counting the number of conditional functions the SPE has performed since the previous execution of a processing function (IFCNT[3:0]). This counter is reset when a new processing function is executed, or when the SPE goes to WFE.
- Providing a comparison history of conditional functions since the previous execution of a processing function (IFHIST[14:0]). For example, if three conditional functions have been executed since the previous execution of a processing function.
  - IFHIST[0] contains 1 if the comparison of the first conditional function was true.
  - IFHIST[1] contains the comparison result of the second conditional function.
  - IFHIST[2] contains the comparison result of the third conditional function.

  IFHIST and IFCNT are stored in the capture register only when the next processing function is executed and if CAPMOD[1:0] = 3.

- Reformat the spe_pfct_pulse, spe_if_pulse, and spe_iftrue_pulse signals for the SNOOP block. Each signal is stretched to a minimum pulse width of five hsp_hclk_core_ck periods.

The figure below shows a simplified block diagram of the trace controller.

Figure 189. Trace controller block diagram



The TSC also provides a clock prescaler (PRESC[4:0]) cascaded with a 14-bit free-running counter (FRCNTR). The prescaler uses the HSP core clock as its timebase (**hsp_hclk_core_ck**). The PRESC[4:0] offers a 15-bit dynamic range, allowing the application to adjust the time granularity according to the desired trace.

The free-running counter and prescaler are enabled with the FRCNTEN bit.

When FRCNTEN is cleared, PRESC[4:0] and FRCNTR are reset, and no longer clocked. To change the prescaler value safely (PRESC[4:0]), the application must first clear FRCNTEN.

When a capture event occurs, the TSC stores the value of various signals in the HSP_CAPDR register:

- Timestamp (TSTAMP[13:0])
- Processing list number (TSKNB[5:0])
- Processing function number (PFCTNB[5:0])
- Number of conditional functions executed since the previous processing function (IFCNF[3:0])
- Comparison of the status history for each conditional function (IFHIST[14:0])

The TSC offers two data capture formats for more flexibility. This format is selected through the FRCNTEN bit.

- When FRCNTEN is cleared, the time-stamp function is disabled, and HSP_CAPDR contains:
  - The processing list number (TSKNB[5:0])
  - The processing function number (PFCTNB[5:0])
  - The conditional function counter (IFCNT[3:0])
  - The conditional function comparison status history (IFHIST[14:0])
- When FRCNTEN is set, the timestamp function is enabled, and HSP_CAPDR contains:
  - The timestamp (TSTAMP[13:0])
  - The processing list number (TSKNB[5:0])
  - The conditional function counter (IFCNT[3:0])
  - A reduced set of the conditional functions comparison status history (IFHIST[7:0])

Note: Having 15 bits for IFHIST allows keeping the history of up to 15 consecutive conditional functions. Having 8 bits for IFHIST keeps the history of up to 8 consecutive conditional functions.

When a capture occurs, the HSP performs the following actions:

- The parameters listed above are stored into the capture register (HSP_CAPDR).
- The flag CAPRDYF is activated (with the **tsc_cap_rdy** signal) to inform the CPU that the capture register has been updated.
- A capture event signal is sent to the TRGO to generate a trigger pulse (see *Section 20.4.15: Trigger outputs (TRGO)*).

PRESC[4:0], FRCNTEN, TSKFLT, and CAPMOD[1:0] are located in the *HSP capture control register (HSP_CAPCR)*.

*Note: PFCTNB[5:0] (**tsc_pfctnb**[5:0]) represents the number of processing functions since the start of the current task. This value is incremented every time a processing function is executed. When the capture event is due to a change of TSKNB[5:0] (**tsk_active**[5:0]), PFCTNB is equal to 0.*

*The **tsc_pfctnb**[5:0] is also available on I/Os with the SNOOP function.*

Figure 190 shows how the TSC works, and which kinds of information can be captured.

Initially, the free-running counter is enabled and CAPMOD is cleared, indicating that the capture is disabled. Then the application clears and sets FRCNTEN to restart the free-running counter.

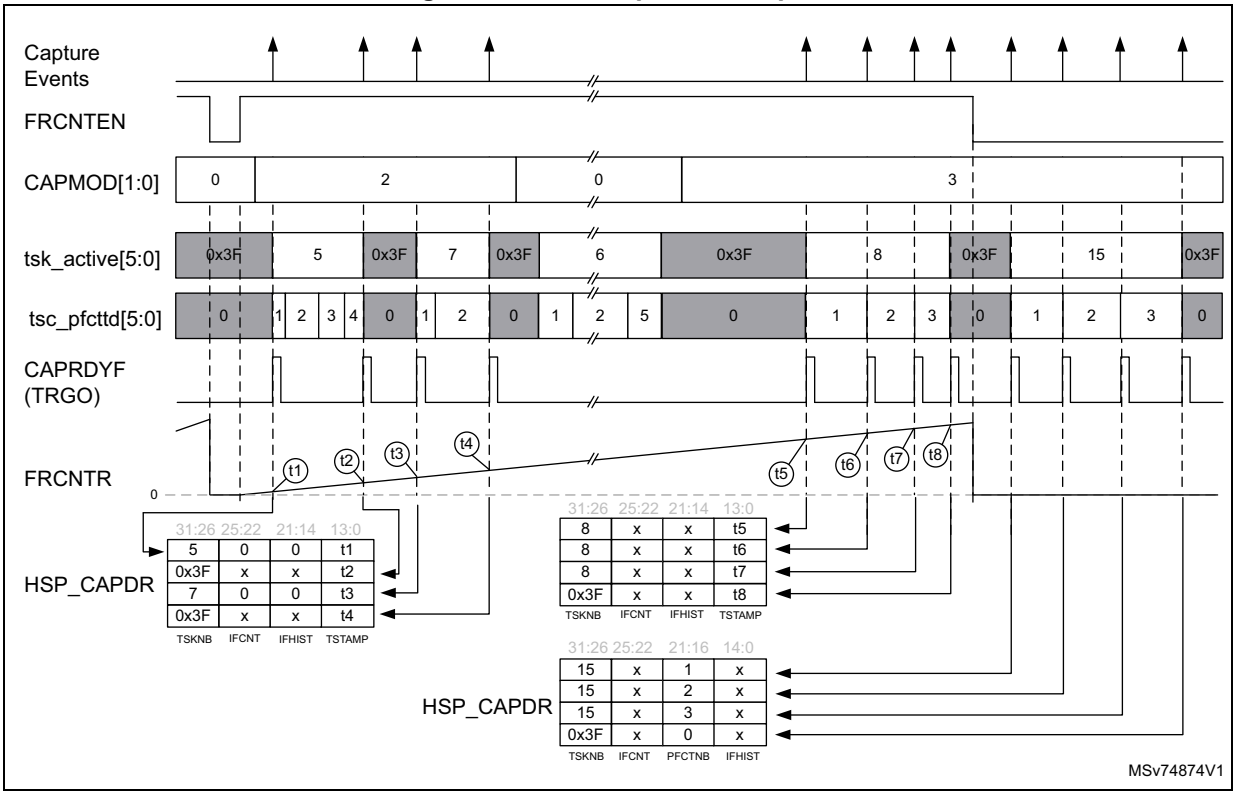
When CAPMOD is set to 2, the TSC stores all transitions of the task number in HSP_CAPCR. The CAPRDYF flag is set every time data are ready in HSP_CAPCR, and cleared when the application reads data. In this mode, IFCNT and IFHYST are not meaningful. When CAPMOD = 2, PFCTNB is cleared.

Later, CAPMOD is cleared and the free-running counter is still working, but the capture interface is disabled.

Next, CAPMOD is set to 3, with FRCNTEN = 1, allowing the TSC to store all processing function transitions with an associated timestamp in HSP_CAPCR. In this case, the amount of captured data may be important. In some cases, an overrun may occur. In this mode, IFCNT and IFHIST are updated according to the content of the processing list.

Finally, if FRCNTEN is cleared, the HSP_CAPDR format is changed and the timestamp is no longer present, but the depth of IFHIST is increased and PFCTNB is added.

Figure 190. TSC capture example 1



Note: IF CAPMOD is set to 2 before the boot of the SPE (BOOTEN = 1), the TSC captures an event, even when the SPE goes to WFE for the first time.

When TSKNB[5:0] = 0x3F and PFCTNB[5:0] = 0, it means that the SPE is in WFE.

When two processing lists are spaced by only one cycle in WFE, the WFE state (event indicating the end of the processing list) is not recorded in HSP_CAPDR, avoiding the generation of a probable overrun situation.

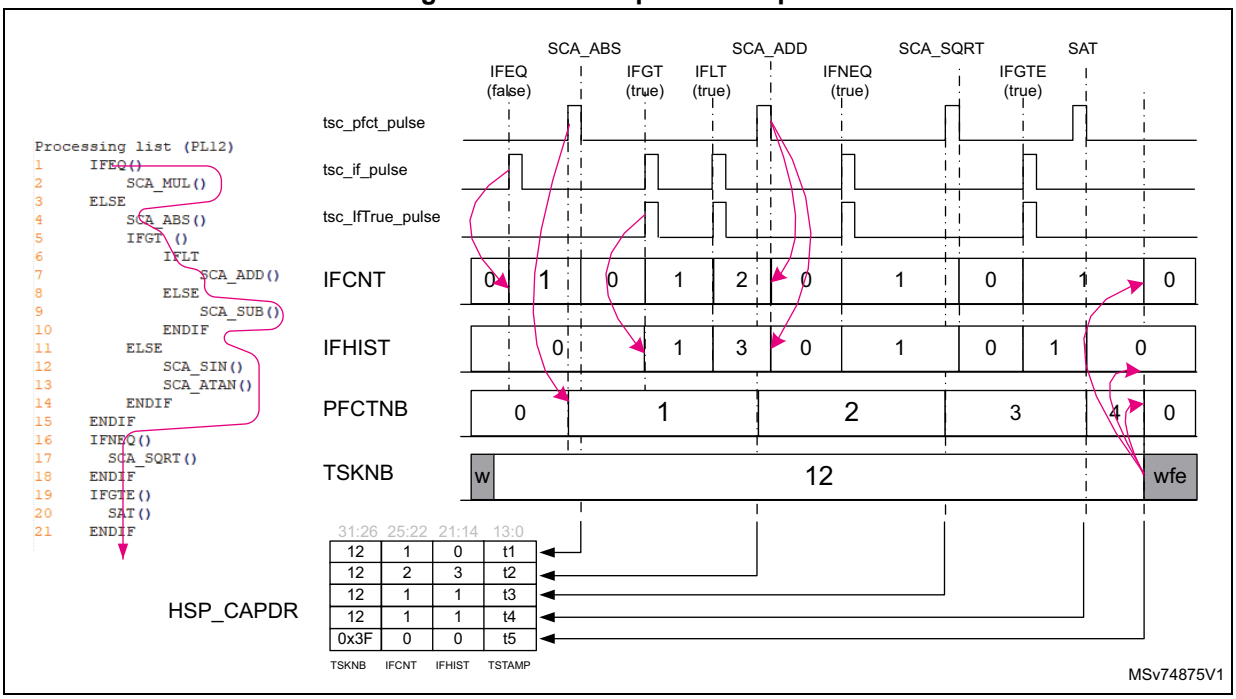
Figure 191 details the case where CAPMOD is set to 3, with a processing list containing conditional functions.

The left side of *Figure 191* shows the (simplified) processing list to be traced, with the red line indicating what has been executed. Processing functions in lines 2, 9, 12, and 13 are not executed.

The first record in HSP_CAPDR shows that one conditional function has been executed before this record. Using the processing list code, the application can deduce that the IFEQ comparison result is false (IFHIST[1] = 0). The ELSE part is then executed, and this first record corresponds to SCA_ABS.

The second record in HSP_CAPDR, at t2, shows that two conditional functions have been executed since t1. Using the processing list code, the application can deduce that after executing SCA_ABS, the IFGT comparison result was true (IFHIST[0] = 1). The ELSE part has not been executed. This is also confirmed by the true IFLT comparison result (IFHIST[1] = 1). The record at t2 corresponds to SCA_ADD, and so on.

Figure 191. TSC capture example 2



If the capture register needs to be written with a new record while the application did not yet read the previous value:

- The TSC activates the overrun flag (CAPOVRF).
- The TSC does not overwrite the capture register. The previous unread value is not corrupted.
- The TSC does not generate a capture event for the TRGO.

The TSC also provides CAPRDYF, asserted when a data is available in the HSP_CAPDR register. This flag is automatically cleared when the CPU reads HSP_CAPDR.

If CAPRDYIE is set in the *HSP events interrupt enable register (HSP_EVT_IER)*, an interrupt can be generated to the CPU.

Note: A TRGO signal can be generated as well. Depending on the product, it allows using the device DMA to read HSP_CAPDR (see Section 20.4.15: Trigger outputs (TRGO)).

Application example: catching the SPE state in case of SPE break

The application must follow this programming sequence:

1. Clear FRCNTEN.
2. Clear CAPMOD[1:0].
3. In the BKITF, enable all sources that are able to freeze the SPE.
4. Set CAPMOD[1:0].

This configuration can be helpful if the application wants to log field issues and identify what the SPE was doing when a break event occurred. The timestamp is not enabled in this example.

Application example: capturing processing function timestamps

The application must follow the programming sequence below:

- If the application intends to generate capture events only for processing functions located in a specific task:
  a) In the TCU, write the task number where the trace must be done in TSKCMP0[5:0] in the *HSP task comparator unit register (HSP_TCUCFGR)*.
  b) Enable the task comparator 0 by setting TC0EN in HSP_TCUCFGR.
  c) In BKITF, set TSKFLT.
- If the application intends to generate capture events every time a new processing function is executed, clear TSKFLT in the BKITF block. It is not necessary to program the TCU block.
- Prevent the processing function transition detector to break the SPE by setting SSEN = 0.
- In the TSC:
  – Clear FRCNTEN.
  – Clear CAPMOD[1:0].
  – Set the PRESC[4:0] to the desired value.
  – Set FRCNTEN.
  – Select the processing functions capture mode (CAPMOD[1:0] = 0b11).

The application can read the captured values in several ways:

- In polling mode, by waiting for CAPRDYF = 1, then reading HSP_CAPDR. This method is simple but not flexible and requests CPU load.
- In interrupt mode, by setting CAPRDYIE to allow the HSP to generate an interrupt every time new data are available. This is a simple and flexible method, but requests CPU load.
- Program the HSP to generate an event on one of the **hsp_trgo**[x] signals. This can be done by setting TRGO0SRC[1:0] to 3, for example, to use the **hsp_trgo_out**[3] signal. Depending on the product capabilities, the **hsp_trgo**[x] signals are connected to the trigger inputs of the device DMA, allowing it to read HSP_CAPDR every time a new timestamp is available. This method is the most efficient for the CPU loading but also for real time.

Note: Performing captures of processing function transitions may cause real-time issues, depending on the traced processing function. The execution time of some processing functions can be down to 10 ***hsp_hclk_core_ck*** *cycles, causing potential overrun if HSP_CAPDR is not read on time.*

Warning: When CAPMOD[1:0] must be modified, it is recommended to clear it before setting it to the desired value.

20.4.17 BRAM conflict counter

The BRAM can be accessed by the following masters:

- SPE
- SoC master devices via HSP slave port SP1

The priority is given to the SPE, but the application may want to check if access requests to the BRAM-A, BRAM-B, or BRAM-AB suffer from contentions, reducing the overall performance. For this purpose, the HSP offers a 32-bit conflict counter incremented every time the MMC detects a contention on BRAM access.

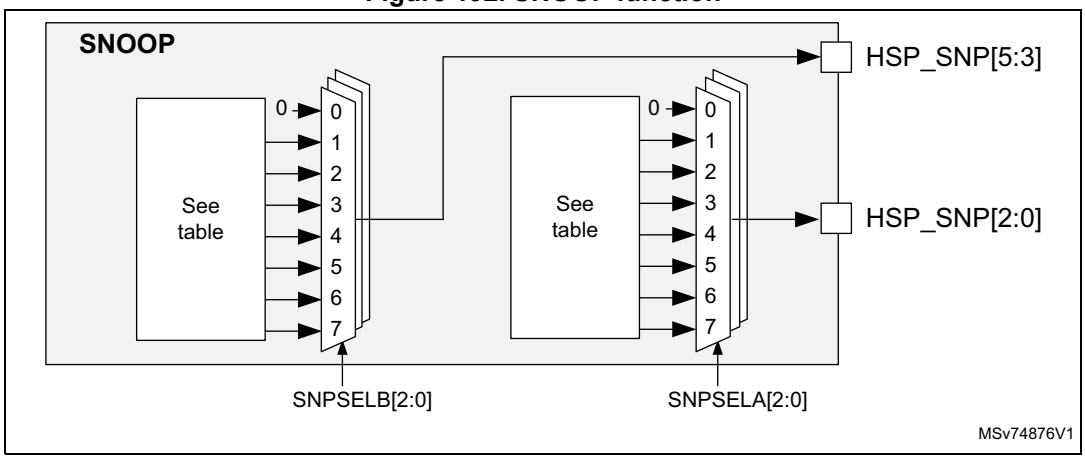
This feature can be enabled or disabled through CCNTREN in the *HSP capture control register (HSP_CAPCR)*. The conflict counter is reset when CCNTREN is disabled.

The conflict counter value can be read at any moment through the *HSP conflict counter register (HSP_CCNTR)*.

20.4.18 Snoop function (SNOOP)

The HSP offers the possibility to observe key signals to ease application tuning. The application can select which set of signals can be observed on the HSP_SNP[5:0] pads through SNPSELB[2:0] and SNPSELA[2:0] in the *HSP SNOOP register (HSP_SNPR)*.

Figure 192. SNOOP function



The table below shows the signal that can be observed through pads.

Table 148. Snoop signal selection

SNPSELx[2:0]	HSP_SNP[5]	HSP_SNP[4]	HSP_SNP[3]	HSP_SNP[2]	HSP_SNP[1]	HSP_SNP[0]
	SNPSELB[2:0]			SNPSELA[2:0]		
0	Disabled	Disabled	Disabled	Disabled	Disabled	Disabled
1	tsk_active[5]	tsk_active[4]	tsk_active[3]	tsk_active[2]	tsk_active[1]	tsk_active[0]
2	prc_evt[25]	prc_evt[24]	prc_evt[23]	-	-	-
3	dcmd_evt	cdeg_evt	hdeg_evt	trgin_evt[0]	buff_evt[0]	adc_evt[0]
4	tsc_pfctnb[5]	tsc_pfctnb[4]	tsc_pfctnb[3]	tsc_pfctnb[2]	tsc_pfctnb[1]	tsc_pfctnb[0]
5	-	-	tsk_cmp[3]	tsk_cmp[2]	tsk_cmp[1]	tsk_cmp[0]
6	tsc_call_pulse	tsc_if_pulse	tsc_iftrue_pulse	evtc_wkup	spe_jump_evt	spe_wfe
7	pe_evt[5]	pe_evt[4]	pe_evt[3]	pe_evt[2]	pe_evt[1]	pe_evt[0]

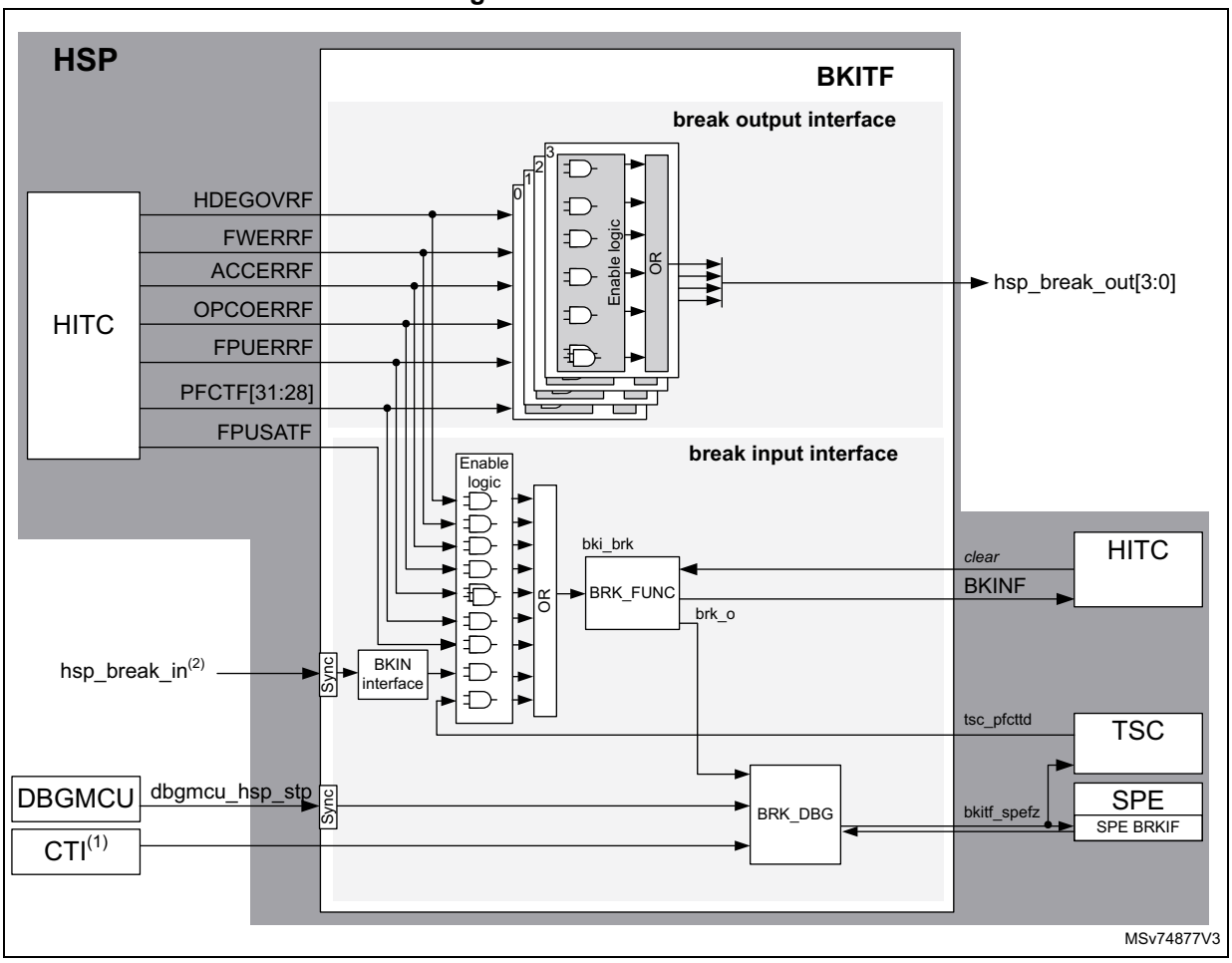
20.4.19 Break interface (BKITF)

The HSP offers a break output interface, providing up to four break output signals (**hsp_brk_out**[3:0]). These signals can be used by the application to stop sensitive peripherals when a critical issue is detected in the HSP.

The HSP also offers a break input interface allowing the application to freeze the SPE in case of internal errors or errors reported by an external block through **hsp_break_in**. When the SPE breaks, it is possible to store in a capture register which processing list the HSP was executing, as well as a timestamp.

In addition, the BKITF allows the step-by-step processing list execution with the processing function granularity for debugging purposes.

Figure 193. Break interface



1. CTI is not present in all products (refer to *Section 20.3: HSP implementation*).
2. hsp_break_in is not always connected; refer to *Table 139: HSP break connections on STM32H7Power* for more details.

HSP break output interface

For each break output signal, it is possible to select one or several flags as the source. The table below lists the possible sources for the break output interface. The control bits listed in the table must be set to allow the corresponding flag to generate a break signal.

Table 149. Flags and signal sources for break function

Source	Description	Control(1)
ACCERRF	Access error flag	ACCEREN
FPUERRF	FPU Computation errors	FPUEREN
OPCOERRF	Illegal operating code	OPCEREN
PFCTF[31:28]	Flags set by processing function (application dependent)	PFCT[31:28]EN
HDEGOVRF	Dedicated software event generator error	HDEGOVEN
FWERRF	Firmware errors	FWEREN
FPUSATF	FPU saturation flag	FSATEN

1. These bits are in the *HSP break output configuration register x (HSP_BKOxCFGR)*.

Note: These flags are asserted by the HSP or firmware code, and cleared by the application (see Table 146: HSP interrupt requests).

The **hsp_break_out**[x] signal is set when at least one of the enabled sources is activated, and goes back to 0 when all enabled sources are inactive.

HSP break input interface

The break input interface can be used to:

- Freeze the SPE
- Generate capture events to the TSC

The freeze of the SPE can be activated by various sources:

- Several error status flags
- A break input signal
- A start of the end of execution of processing functions
- A request from the DBGMCU
- A request from CTI (if available in the product)

Table 150 lists the possible sources for the break input interface. The control bits must be set to allow the corresponding flag to generate a break signal. These control bits are in the *HSP break input configuration register (HSP_BKICFGR)*.

Note: The BKINF flag is not asserted by all available break sources.

Table 150. Flags and signal sources for break input and debug functions

Break sources	Description	HSP control (1)	Affected flag	Affected functions
ACCERRF	Access error flag	ACCEREN	BKINF	SPE freeze
FPUERRF	FPU Computation errors	FPUEREN		
OPCOERRF	Illegal operating code	OPCEREN		
PFCTF[31:28]	Flags set by processing functions	PFCT[31:28]EN		
HDEGOVRF	Dedicated software event generator error	HDEGOVEN		
FWERRF	Firmware errors	FWEREN		
FPUSATF	FPU saturation flag	FSATEN		
tsc_pfctdt	Processing function transition detection	SSEN		
DBGMCU	HSP freeze when CPU breaks	-(2)	-	SPE freeze
CTI	CoreSight CTI	-		

1. These bits are located in the *HSP break input configuration register (HSP_BKICFGR)*, except for DBGMCU and CTI sources.
2. DBGMCU has a bit to control this function (see the DEBUG section of the reference manual).

The SPE freeze is controlled by the **bkitf_spefz** signal. When **bkitf_spefz** is asserted, the SPE immediately stops fetching code and remains frozen. The HSP_CTRL is still accessible, allowing the application to access all HSP registers and memories (conditioned by the REGLCK bit).

The SPE can resume the execution if all enabled break sources are inactive and the corresponding flags are cleared.

In addition, the moment the SPE is frozen, a capture event is sent to the TSC to store the time-stamp, the task number, and the processing function number.

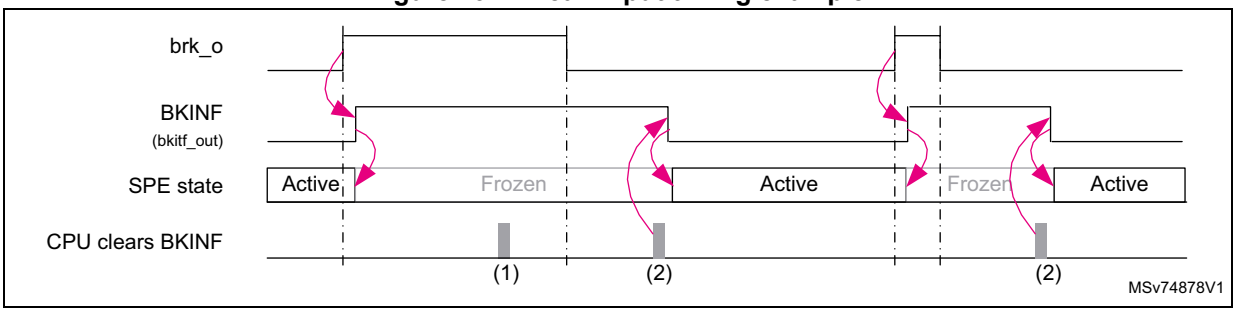
If the break is generated by BRK_FUNC, the BKINF is asserted as well. As shown in *Figure 194*, when the application resets the BKINF flag, if the break condition is still present at the input, the BKINF flag is not reset (#1 in the figure). If the break condition is no longer present, the CPU can clear the BKINF flag, and the SPE can resume the execution (#2 in the figure).

When the SPE is in break, the application can decide on one of the following actions:

- Resume the SPE execution: the application clears all enabled break sources and clears the BKINF (by setting BKINC).
- Restart the complete application: the application maintains the SPE frozen by not resetting the BKINF and performs an HSP reset through the RCC control bits.

The break input function is configured through the *HSP break input configuration register (HSP_BKICFGR)*.

Figure 194. Break input timing example



20.5 Debug

The HSP offers the following debug options:

- Freeze the SPE according to some error status flags.

  This can be helpful to find the processing function that is causing the error (see *Application example: SPE freeze on specific error conditions*).
- Freeze the SPE every time the SPE starts the execution of a new processing function (see *Application example: SPE step-by-step debug*).
- Capture a timestamp every time a new processing function is executed (see *Application example: capturing processing function timestamps*).
- Freeze the SPE when the CPU is halted by using a control bit located in the DBGMCU block.

Application example: SPE freeze on specific error conditions

This feature can be used in applications requesting the SPE to stop immediately in case of errors inside or outside the HSP. It can also be used for debugging purposes, to find the processing function generating an unexpected error, for example.

To use this function, the HSP must be programmed as follows:

- Define which flags or signals generate an SPE break (see *Table 150: Flags and signal sources for break input and debug functions*).
- If CAPMOD[1:0] of the TSC block is set to 0b01, when the SPE breaks, the application finds the timestamp, the last task, and the processing function number in the HSP_CAPDR register.

When the SPE breaks due to error detection, the application can decide:

- To resume the SPE execution.
- To restart the complete application.

Application example: SPE step-by-step debug

This feature can be helpful for debugging purposes to analyze data buffers after the execution of each processing function. It is possible to perform this step-by-step debugging only for processing functions located in a specific task, or for any processing function.

To use this feature, the HSP must be programmed as follows:

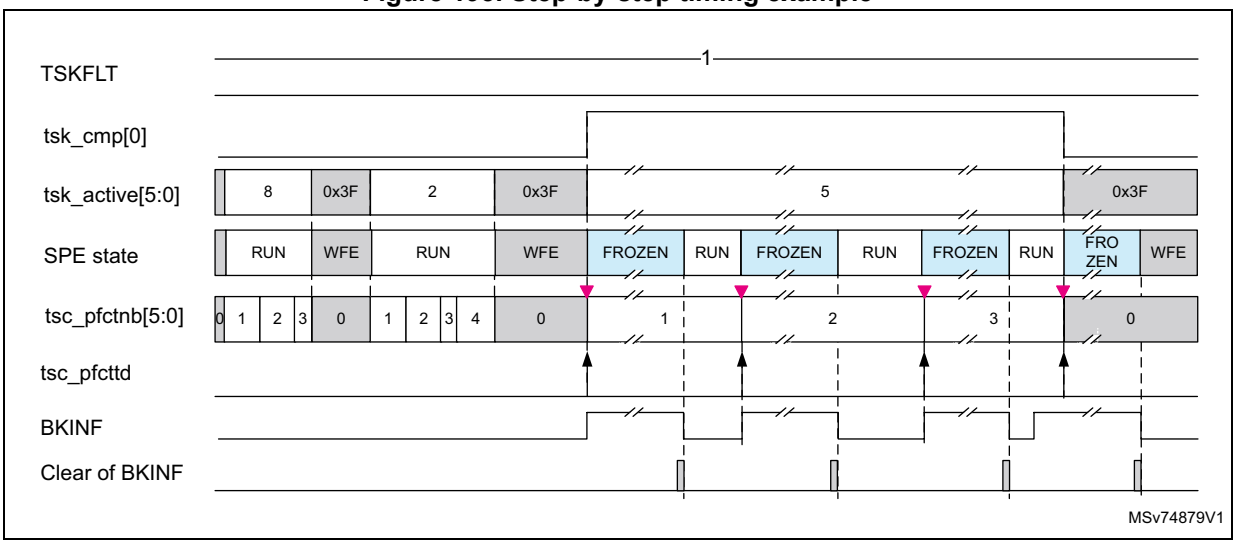
- If the application intends to break the SPE only when a specific task is executed:
  - In the TCU, write the task number where the step-by-step feature must be activated in TSKCMP0[5:0] of the *HSP task comparator unit register (HSP_TCUCFGR)* and enable the task comparator by setting TC0EN in HSP_TCUCFGR.
  - Set the TSKFLT bit of the HSP_CAPCR register.
- If the application intends to break the SPE every time a new processing function is executed, clear TSKFLT. No need to program the TCU block.
- Allows the step-by-step feature to break the SPE by setting SSEN = 1.
- If the capture mode of the TSC is configured to perform a capture every time the SPE is frozen (CAPMOD[1:0] = 0b01), when the SPE breaks, the application finds the task under execution, the processing function number, and a timestamp in HSP_CAPDR.

When the SPE is in break, the application can check buffer contents and resume the SPE execution. To resume the SPE execution, the application must clear the BKINF flag by setting BKINC in the *HSP error interrupt clear register (HSP_ERR_ICR)*. If additional flags or signals are able to generate a break, all enabled sources must be cleared to resume the SPE.

The CPU can be broken when the SPE breaks (see *Section 20.5.1: Register protection*).

In the figure below, TSKCMP0[5:0] is set to 5. The SPE runs normally until it starts the execution of task number 5. At the start of the execution of task number 5, the SPE breaks, allowing the application to dump the data before execution of processing function number 1. Then the application clears the BKINF flag, and the SPE executes processing function number 1 and breaks again at the start of processing function number 2, allowing the application to dump data, and so on.

Figure 195. Step-by-step timing example



20.5.1 Register protection

The HSP embeds some hardware protection to prevent unexpected changes to register contents when the HSP is activated. *Table 151* shows the list of write-protected and unprotected fields. The register protection is mainly handled by the REGLCK bit.

Note: When the application sets the REGLCK bit, it is no longer possible to clear it by writing this bit to 0. The HSP must be reset.

Table 151. List of locked fields

Register	Unprotected fields	Write-protected fields	Write protection condition
HSP control register (HSP_CR)	BOOTEN REGLCK RSTREQ SPECGDIS MMCCGDIS CTRLCGDIS DCMDEVEN	-	BOOTEN = 1
HSP event enable register (HSP_EVTENR)	-	All	REGLCK = 1
HSP event synchronization enable register (HSP_ESYNCENR)	-	All	REGLCK = 1
HSP event synchronization source register 0 (HSP_ESYNC0SRCR) *HSP event synchronization source register 1 (HSP_ESYNC1SRCR)* *HSP event synchronization source register 2 (HSP_ESYNC2SRCR)*	-	All	REGLCK = 1 or EVTENx = 1
HSP event source register 0 (HSP_EVTSRC0R) *HSP event source register 1 (HSP_EVTSRC1R)* *HSP event source register 2 (HSP_EVTSRC2R)*	-	All	REGLCK = 1 or EVTENx = 1
HSP BUFF configuration register (HSP_BUFFCFGR)	-	All	REGLCK = 1
HSP BUFF configuration register (HSP_BUFFCFGR)	-	All	REGLCK = 1 or BUFFEN = 1
HSP TRGIN configuration register (HSP_TRGINCFGR) *HSP TRGIN input selection register 0 (HSP_TRGINSELR0)* *HSP TRGIN input selection register 1 (HSP_TRGINSELR1)* *HSP TRGIN input selection register 2 (HSP_TRGINSELR2)*	-	All	REGLCK=1 or TRGIxEN = 1
HSP TRGO configuration register (HSP_TRGOCFGR)	-	All	REGLCK = 1 or TRGOEN = 1
HSP task comparator unit register (HSP_TCUCFGR)	-	All	REGLCK = 1
HSP task overlap control register (HSP_TOVLPCR)	-	All	REGLCK = 1
HSP break output configuration register x (HSP_BKOxCFGR)	-	All	REGLCK = 1
HSP break input configuration register (HSP_BKICFGR)	-	All	REGLCK = 1

20.6 HSP firmware description

The HSP is built around a dedicated high-performance signal processor engine (SPE) running the firmware embedded in the CROM and the CRAM. The CROM mainly contains a set of optimized signal processing functions (libraries), and the CRAM contains the scheduler part of the HSP.

These signal processing functions are called “processing functions”, and the application can record several lists of processing functions to be executed upon activation of specific events. These lists of processing functions are called “processing lists”.

Events generated by peripherals trigger tasks, and each task can contain a single processing list.

This section gives an overview of the way that the application can perform the various operations supported by the HSP, such as running a direct command or recording a processing list.

This document does not provide details on processing functions; refer to the user manual *HSP processing functions on STM32 MCUs* (UM3360).

20.6.1 Processing list mapping

Table 152 shows the processing list mapping. Processing lists 0 to 36 are dedicated to recording application processing lists.

Processing lists 37, 38, and 39 are built-in processing lists, not available to the application.

- The processing list number 37, named PLSUP, is responsible for several control tasks, such as recording processing lists.
- The processing list number 38, named PLDCMD, is used to run direct commands when the application wants to use the HSP in accelerator mode.
- The processing list number 39, named PLFAIL, is triggered every time the application tries to execute a processing list with a number higher than 38 or when the application wants to execute a nonexiting processing list.

The built-in processing lists cannot be modified by the application.

After an HSP reset, processing lists 0 to 36 are routed to PLFAIL, to handle the activation of non-exiting processing lists. Every time the application records a new processing list, the link to PLFAIL is replaced by the link to the new processing list.

Several blocks can trigger a processing list: TRGIN, STREAM, CSEG, HSEG, CDEG, HDEG, DCMD.

Table 152 shows that:

- Processing list number 0 can only be triggered by HDEG.
- Processing lists numbers 1 to 22 can be triggered by TRGIN, STREAM, CSEG, HSEG, CDEG, and HDEG.
- Processing lists numbers 23 to 36 can be triggered by CDEG and HDEG
- Processing list number 37 can be triggered by CDEG.
- Processing list number 38 can be triggered by DCMD.

CDEG always has the priority 27 (lowest); HDEG always has the priority 0 (highest); DCMD always has the priority 26.

TRGIN, STREAM, CSEG, and HSEG always have the priority of the processing number they are activating. For example, if STREAM is programmed to trigger processing list number 12, then its priority is 12.

Table 152. Processing list mapping

Peripheral generating events	-	Processing list number	Processing list usage
CDEG, HDEG	→	0	Application PL or jump to PLFAIL
TRGITF, STREAM, CSEG, HSEG, CDEG, HDEG	→	1	Application PL or jump to PLFAIL
	→	...	...
	→	22	Application PL or jump to PLFAIL
CDEG, HDEG	→	23	Application PL or jump to PLFAIL
	→	...	...
	→	36	Application PL or jump to PLFAIL
CDEG	→	37	PLSUP
DCMD	→	38	PLDCMD
-	→	39	PLFAIL

20.6.2 Sending commands to the HSP

The application must send commands to the HSP in the following cases:

- To complete the HSP initialization phase by sending the FW_INIT command.
- When a processing list must be programmed.
- When a direct command must be executed.
- When the application wants to erase all processing lists.

Sending the FW_INIT command

The firmware initialization command (FW_INIT) is sent only once during the initialization phase to give to the HSP firmware the base address of the BRAMAB, and to activate or not the performance monitor. This base address may differ from a product to another.

To send the FW_INIT command, the application must do the following:

1. Wait until the HSP is ready to accept the FW_INIT command, signified by BSTAT[3:0] = 1.
2. Send the FW_INIT command as follows:
   a) Check that MSGB is free (C2HSEM is cleared).
   b) Write the base address of BRAM-AB in HSP_PARAMR0. This address changes from one product to another, so the application needs to provide this information to the HSP.
   c) Clear HSP_PARAMR1 to disable the performance monitor, and set it to enable it.
   d) Write the command ID of the FW_INIT command in HSP_C2HMSGDR.
   e) Set C2HSEM.
3. Check for the completion of the command execution (H2CSEM is set).
4. Read the command status in HSP_H2CMSGDR. If the status differs from 0, the command has not been processed properly.
5. Clear H2CSEM.

Sending commands to the supervisor

The PLSUP built-in processing list activates the supervisor function. The supervisor can be used to perform various tasks including the processing list recording. The supervisor can be activated at any moment by writing 37 in the *HSP CPU dedicated event generation register (HSP_CDEGR)*.

The supervisor event has the lowest priority. If it is activated while other active events are pending, it is served when all other events are served.

To send a command to the supervisor, the application must:

1. Activate the supervisor.
2. Use the MSGB and HSP_PARAMRx to send commands.

The recommended procedure to send a command to the supervisor is detailed below:

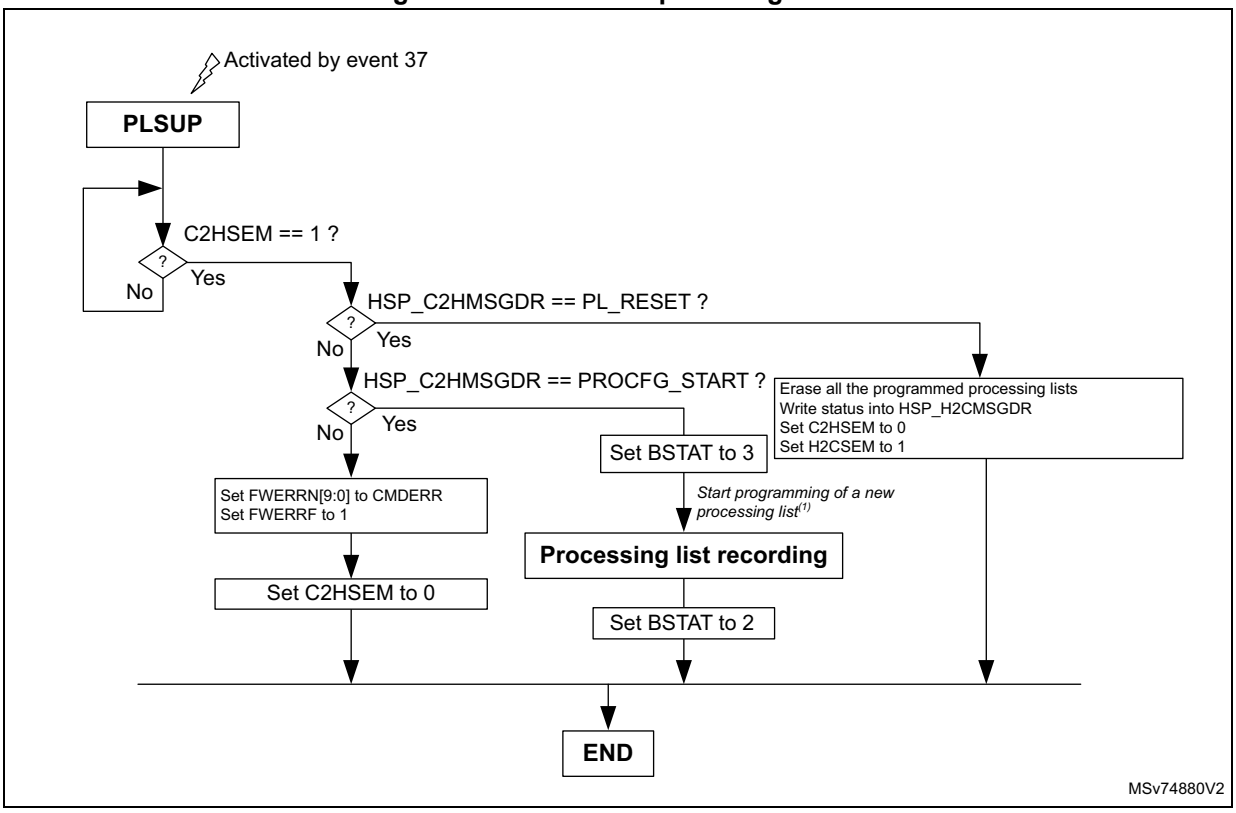
Sending a command to the supervisor

1. If not yet enabled, set CDEGEN in the HSP_IFTENR register.
2. Activate (if not done yet) the supervisor as follows:
   a) Check that CDEG is free (CDEGBSY is cleared).
   b) Write 37 in the HSP_CDEGR register.
3. Check that MSGB is free (C2HSEM is cleared).
4. Write the command parameters in the HSP_PARAMRx registers.
5. Write the command ID in the HSP_C2HMSGDR register.
6. Set C2HSEM in the HSP_C2HSEMR register.
7. Wait for the completion of the command execution (H2CSEM is set).
8. Read the HSP_H2CMSGDR register to get the command status.
9. Clear H2CSEM.

As shown in the figure below, the supervisor can perform the following operations:

- Erase all existing processing lists.
- Program new processing lists.

Figure 196. PLSUP simplified algorithm



1. Refer to *Figure 197: Processing list programming algorithm*.

20.6.3 Recording processing lists

At any time, the application can add a new processing list to the HSP by activating the supervisor:

1. Send the PROCFG_START command as follows:
   a) Activate the supervisor by writing 37 in HSP_CDEGR (this action activates the supervisor, if not done yet).
   b) Check that C2HSEM is cleared.
   c) Write the processing list number in the HSP_PARAMR0 register.
   d) Write the PROCFG_START command ID in the HSP_C2HMSGDR register.
   e) Set C2HSEM in the HSP_C2HSEMR register.
   f) Wait for H2CSEM to be set, read the command status, and clear H2CSEM.
2. Send the command to record the first processing function as follows:
   a) Check that C2HSEM is cleared.
   b) Write the processing function parameters in the HSP_PARAMRx registers.
   c) Write the processing function command ID in the HSP_C2HMSGDR register.
   d) Set C2HSEM in the HSP_C2HSEMR register.
   e) Wait for H2CSEM to be set, read the command status, and clear H2CSEM.
3. Record the wanted sequence by sending commands as explained in step 2.
4. Write the PROCFG_END command to exit the supervisor:
   a) Check that C2HSEM is cleared.
   b) Write the PROCFG_END command ID in the HSP_C2HMSGDR register.
   c) Set C2HSEM in the HSP_C2HSEMR register.
   d) Wait for H2CSEM to be set, read the command status, and clear H2CSEM.

The number and format of each processing function parameter are described in the user manual *HSP processing functions on STM32 MCUs* (UM3360).

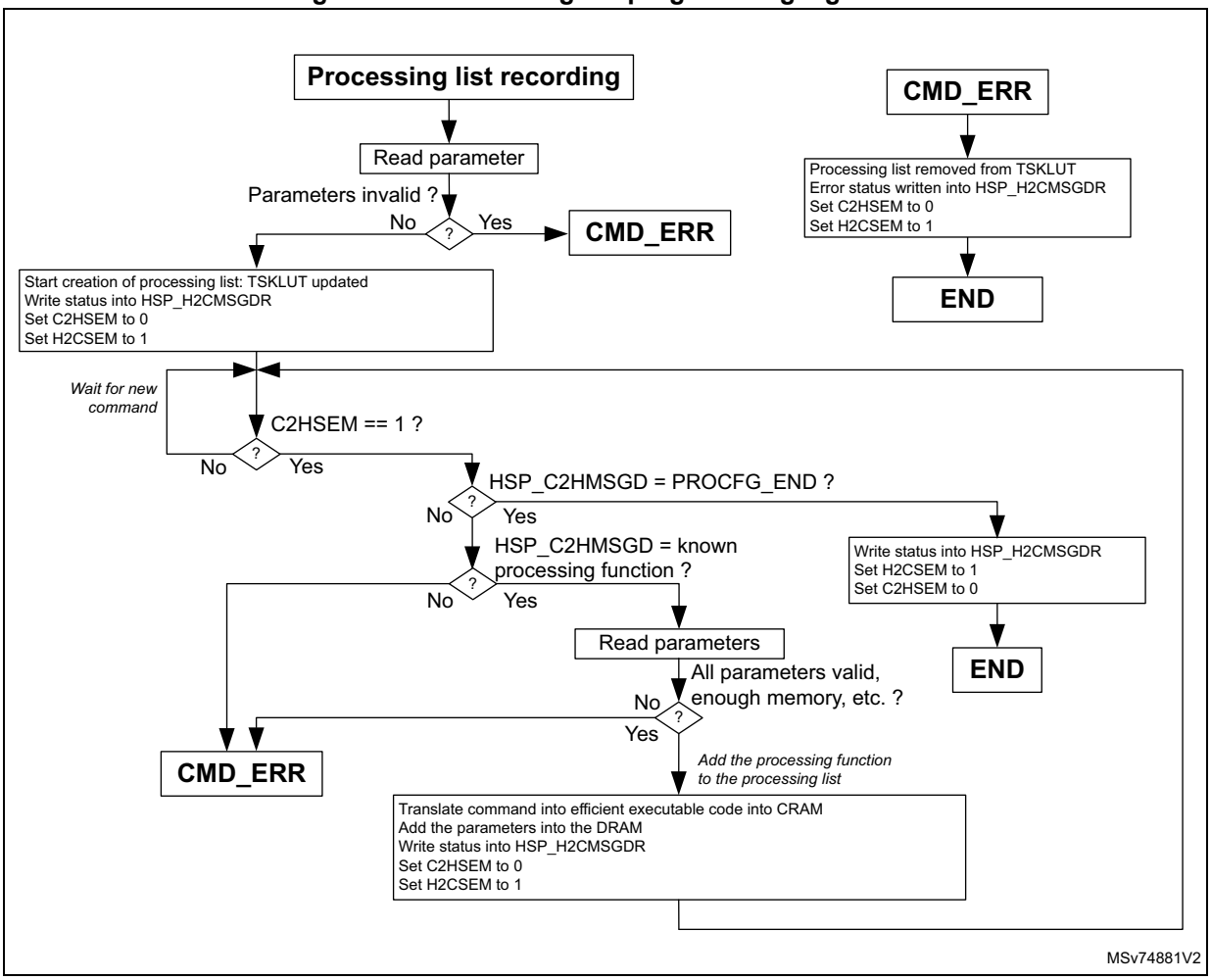
For each command, the HSP firmware sends back a status with HSP_H2CMSGDR.

Figure 197 shows the processing list recording sequence:

1. When the PROCFG_START command is recognized, the supervisor enters into this processing list programming sequencer.
2. BSTAT[3:0] indicates when the supervisor is programming a processing list.
3. When an error is detected, processing list programming is aborted. The processing list address is removed from TSKLUT. The command status reported by the command provides information about the error root cause.

As soon as a processing list is programmed, if its associated event is enabled, activated, and selected by the EVTC, the processing list is executed.

Figure 197. Processing list programming algorithm



20.6.4 Executing direct commands (accelerator mode)

At any time, the application can ask the HSP to execute a processing function as a direct command. If the HSP is executing processing lists, the execution of the direct command is delayed until the EVTC grants the request.

The direct commands are mainly used to execute processing functions dedicated to vector operations. In addition, the processing functions executed in immediate mode can only work with data from BRAM.

The direct command execution is performed through the DCMD and HSP_PARAMRx registers.

The list below provides the recommended sequence to run a direct command. The number of arguments and pointers depend on the command executed (refer to the HSP firmware commands in the user manual *HSP processing functions on STM32 MCUs* (UM3360).

The DCMD block is enabled by default, but can be disabled:

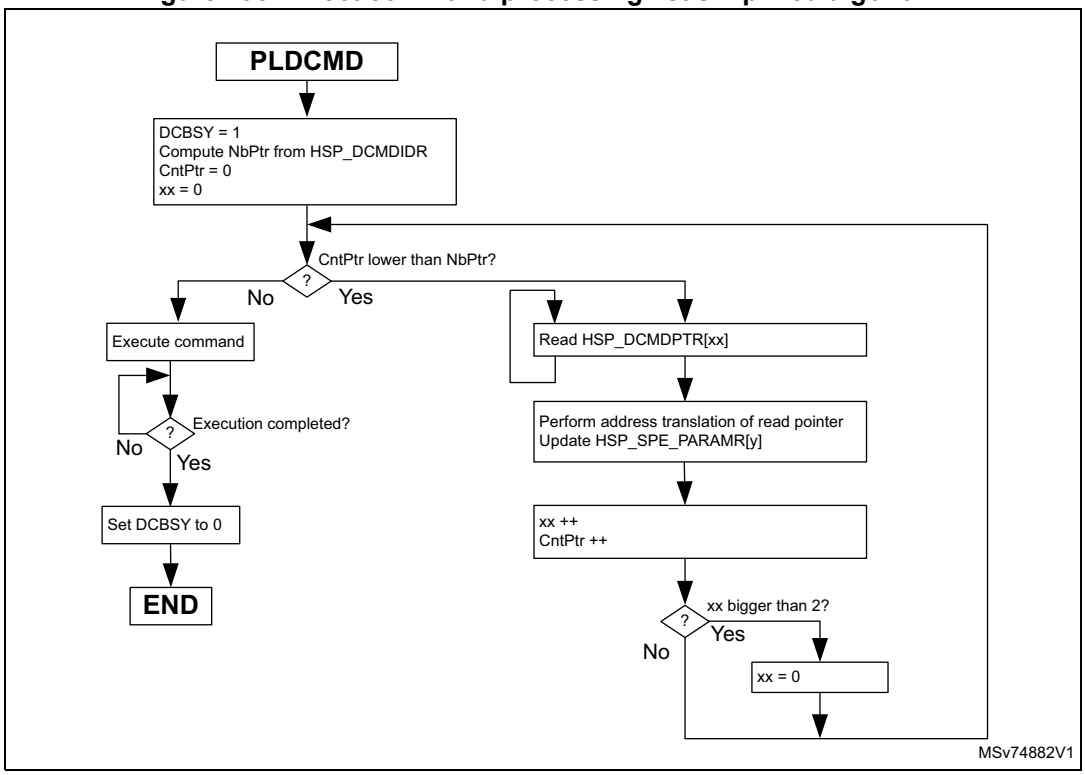
1. Write the command ID of the command to be executed in th *HSP DCMD command ID register (HSP_DCMDIDR)*.
2. Write all parameters except the pointers in HSP_PARAMRx.

3. Write the first pointer in HSP_DCMDPTR0.
4. Write the second pointer in HSP_DCMDPTR1 (if needed).
5. Write the third pointer in HSP_DCMDPTR2 (if needed).
6. Write the fourth pointer as follows (if needed):
   a) Wait for PRTF0 = 0.
   b) Write the fourth pointer in HSP_DCMDPTR0.
7. Write the fifth pointer as follows (if needed):
   – Wait for PRTF1 = 0.
   – Write the fifth pointer in HSP_DCMDPTR1.
8. And so on
9. Wait for command completion by checking that DCBSY = 0.

Figure 198 describes the PLDCMD built-in processing function. The SPE is expecting from the CPU to program the HSP as described above. To reduce overhead, the PLDCMD task does not perform a lot of checks. The application must properly program the direct command interface.

Note: ST provides a set of functions making the HSP interface transparent for the user.

Figure 198. Direct command processing list simplified algorithm



20.6.5 Data buffer handling

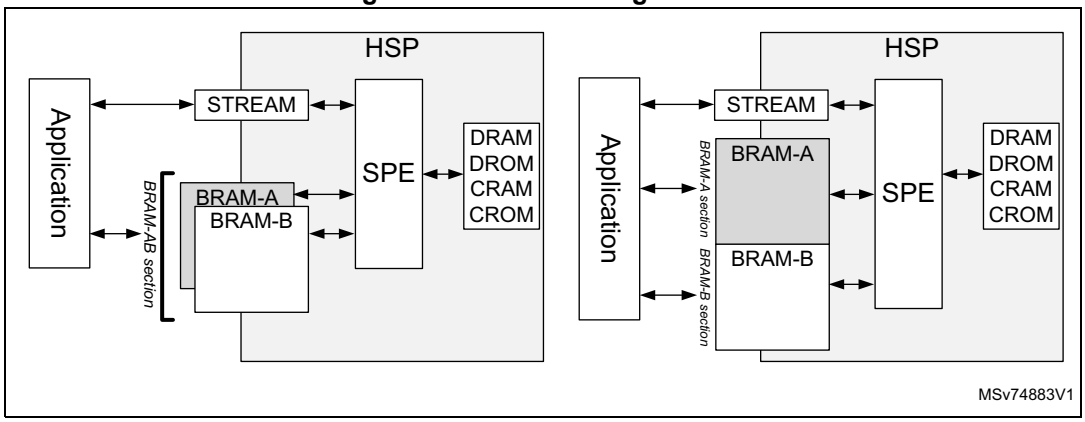
The HSP working buffers are located in the BRAM. Data exchanges between the HSP and the application can use the following interfaces:

- BRAM-A, BRAM-B, or BRAM-AB
- STREAM

Depending on the executed processing function, sharing data through the BRAM must be done using the BRAM-AB section or the BRAM-A and BRAM-B sections. Refer to *Table 153: List of processing functions* for more details.

Using the BRAM-AB section means that the content of BRAM-A must be the same as BRAM-B.

Figure 199. Data sharing models



20.6.6 Performance monitoring

In addition to the TSC block, the HSP also offers a performance monitoring tool, allowing the application to measure the execution time of a processing list.

This function is activated by setting the HSP_PARAM1R register (PERFEN) during the execution of the FW_INIT command.

When activated, a cycle counter value can be read in the last DRAM location after the completion of each processing list. The value is given in HSP core clock cycles (**hsp_hclk_core_ck**). Note that the application cannot read this cycle counter if CDRLCK[1:0] is set to 3.

The cycle counter is cleared every time the HSP starts the execution of a processing list.

Note: The PL_RESET command also allows the enabling or disabling of the performance monitor.

20.6.7 Processing functions summary

The table below gives a summary of all commands that the application can use to program processing functions into processing lists, or to execute them as direct commands. All the commands are described in detail in the user manual *HSP processing functions on STM32 MCUs* (UM3360).

In the table below:

- The column “Type” indicates if the function can be executed immediately in accelerator mode (ACC) or included in a processing list (SEQ).
- The column “STREAM” indicates if input or output data can be taken from the STREAM interface: “Y” = yes, “y/n” = yes for sequencer and no for accelerator mode, “-” means no.
- The column “BRAM” indicates which BRAM section must be used to exchange data: “DP” means that the BRAM-AB section must be used, “SP” means that the BRAM-A, and/or BRAM-B sections must be used.

Table 153. List of processing functions

Processing function mnemonic	Description	Type	STREAM	BRAM
PROCFG_START	Starts the creation of a processing list	SEQ	-	-
PROCFG_END	Ends the creation of a processing list	SEQ	-	-
PL_RESET	Processing list reset	SEQ	-	-
Filter processing functions on real numbers				
BIQDF1_CASC	Cascaded biquad direct form 1 filtering	SEQ, ACC	y/n	DP
BIQDF2T_CASC	Cascaded biquad direct form 2 transposed filtering	SEQ, ACC	y/n	DP
FIR	Optimal FIR filtering	SEQ, ACC	y/n	DP
FIRDEC	FIR filtering with decimation	SEQ, ACC	-	DP
FIRLMS	FIR filtering with adaptive LMS algorithm	SEQ, ACC	y/n	DP
IIRDF1	Optimal IIR filtering using the direct form 1	SEQ, ACC	y/n	DP
IIR_LATTICE	IIR filtering using the Lattice form	SEQ, ACC	y/n	DP
IIR_3P3Z	3rd order IIR filter with saturation and gain adjust	SEQ	Y	DP
IIR_2P2Z	2nd order IIR filter with saturation and gain adjust	SEQ	Y	DP
FLTBANK	Frequency filtering using internal filter banks	SEQ, ACC	-	DP
CONV	Convolution between two vectors	SEQ, ACC	-	DP
CORR	Correlation between two vectors	SEQ, ACC	-	DP
GET_FLTSTATE	Gets the filter state for single stage filters	ACC	-	DP
GET_FLTSTATE_BIQ	Gets the filter state for biquads	ACC	-	DP
SET_FLTSTATE	Sets the filter state for single stage filters	ACC	-	DP
SET_FLTSTATE_BIQ	Sets the filter state for biquads	ACC	-	DP
RST_FLTSTATE	Resets the filter state.	ACC	-	-
Vector processing functions on real				
VECT_ABS	Element-wise absolute value	SEQ, ACC	-	DP
VECT_ABSMAX	Maximum absolute value of a vector	SEQ, ACC	-	DP

Table 153. List of processing functions (continued)

Processing function mnemonic	Description	Type	STREAM	BRAM
VECT_ADD	Element-wise addition	SEQ, ACC	-	DP
VECT_ATAN2	Element-wise arc tangent 2	SEQ, ACC	-	DP
VECT_AVG	Average value	SEQ, ACC	-	DP
VECT_COPY	Vector copy	SEQ, ACC	-	DP
VECT_COS	Element-wise cosine	SEQ, ACC	-	DP
VECT_DEC	Vector decimation	SEQ, ACC	-	DP
VECT_DIV	Element-wise division	SEQ, ACC	-	DP
VECT_DOTP	Dot product	SEQ, ACC	-	DP
VECT_EXP	Element-wise exponential	SEQ, ACC	-	DP
VECT_EXP10	Element-wise 10^x	SEQ, ACC	-	DP
VECT_FTOI	Conversion of float32 to 32-bit signed integers	SEQ, ACC	-	DP
VECT_FTOU	Conversion of float32 to 32-bit unsigned integers	SEQ, ACC	-	DP
VECT_ITOF	Conversion of signed 32-bit integers to float32	SEQ, ACC	-	DP
VECT_I24TOF	Conversion of signed 24-bit integers to float32	SEQ, ACC	-	DP
VECT_Q31TOF	Conversion of signed Q31 to float32	SEQ, ACC	-	DP
VECT_FTOQ31	Conversion of float32 to signed Q31	SEQ, ACC	-	DP
VECT_LN	Element-wise natural logarithm	SEQ, ACC	-	DP
VECT_LOG10	Element-wise logarithm base 10	SEQ, ACC	-	DP
VECT_MAX	Search for value and position of the biggest element	SEQ, ACC	-	DP
VECT_MIN	Search for value and position of the smallest element	SEQ, ACC	-	DP
VECT_MUL	Element-wise product	SEQ, ACC	-	DP
VECT_MULCOS	Element-wise product with a cosine	SEQ, ACC	-	DP
VECT_MULSIN	Element-wise product with a sinus	SEQ, ACC	-	DP
VECT_OFFSET	Adds an offset to each element	SEQ, ACC	y/n	DP
VECT_OFFSETI	Adds an immediate offset to each element	SEQ, ACC	-	DP
VECT_RMS	Root mean square value	SEQ, ACC	y/n	DP
VECT_SIN	Element-wise sinus	SEQ, ACC	-	DP
VECT_SINCOS	Element-wise sinus and cosine	SEQ, ACC	-	DP
VECT_SCALE	Multiplies each element by a value	SEQ, ACC	y/n	DP
VECT_SCALEI	Multiplies each element by an immediate value	SEQ, ACC	-	DP
VECT_SET	Sets a value to each element	SEQ, ACC	y/n	DP

Table 153. List of processing functions (continued)

Processing function mnemonic	Description	Type	STREAM	BRAM
VECT_SETI	Sets an immediate value to each element	SEQ, ACC	y/n	DP
VECT_SQRT	Element-wise square-root	SEQ, ACC	-	DP
VECT_SUB	Element-wise subtraction	SEQ, ACC	-	DP
VECT_UTOF	Conversion of unsigned 32-bit integer to float32	SEQ, ACC	-	DP
VECT_ZINS	Interpolation by inserting zeros	SEQ, ACC	-	DP
Processing functions operating on complex				
CMPLX_CONJ	Element-wise conjugate	SEQ, ACC	-	DP
CMPLX_DOTP	Dot product	SEQ, ACC	-	DP
CMPLX_MAG	Magnitude	SEQ, ACC	-	DP
CMPLX_MAGSQR	Magnitude squared	SEQ, ACC	-	DP
CMPLX_MUL	Element-wise product	SEQ, ACC	-	DP
CMPLX_MULEXP	Element-wise product with an exponential complex	SEQ, ACC	-	DP
CMPLX_RMUL	Element-wise product of a complex with a real vector	SEQ, ACC	-	DP
CMPLX_FTOQ15	Conversion of float32 to Q15 format	SEQ, ACC	-	DP
CMPLX_Q15TOF	Conversion of Q15 to float32 format	SEQ, ACC	-	DP
Scalar processing functions on real				
SCA_ABS	Absolute value	SEQ	Y	DP
SCA_ADD	Addition	SEQ	Y	DP
SCA_ATAN2	Arc tangent 2	SEQ	Y	DP
SCA_CNTCMP	Comparison to a threshold, with a counter	SEQ	-	DP
SCA_COS	Cosine	SEQ	Y	DP
SCA_CLARKE	Clarke transform	SEQ	-	DP
SCA_DIV	Division	SEQ	Y	DP
SCA_EXP	Exponential	SEQ	Y	DP
SCA_EXP10	10^x	SEQ	Y	DP
SCA_FATAN2	Low-precision Arc tangent 2	SEQ	Y	DP
SCA_FCOS	Low-precision cosine	SEQ	Y	DP
SCA_FSIN	Low-precision sinus	SEQ	Y	DP
SCA_FTOI	Conversion of a float32 to signed 32-bit integer	SEQ	-	DP
SCA_FTOQ31	Conversion of a float32 to Q31	SEQ	-	DP
SCA_FTOU	Conversion of a float32 to unsigned 32-bit integer	SEQ	-	DP

Table 153. List of processing functions (continued)

Processing function mnemonic	Description	Type	STREAM	BRAM
SCA_I24TOF	Conversion of a signed 24-bit integer to float32	SEQ	-	DP
SCA_ITOF	Conversion of a signed 32-bit integer to float32	SEQ	-	DP
SCA_Q31TOF	Conversion of a Q31 to float32	SEQ	-	DP
SCA_ICLARKE	Inverses Clarke transform	SEQ	-	DP
SCA_IPARK	Inverses Park transform	SEQ	Y	DP
SCA_LN	Natural logarithm	SEQ	Y	DP
SCA_LOG10	Logarithm base 10	SEQ	Y	DP
SCA_MUL	Product	SEQ	Y	DP
SCA_PARK	Park transform	SEQ	Y	DP
SCA_PID	PID operation	SEQ	Y	DP
SCA_RAMP	Performs a ramp-up or ramp-down	SEQ	-	DP
SCA_SIN	Sinus	SEQ	Y	DP
SCA_SINCOS	Sinus and cosine	SEQ	Y	DP
SCA_SET	Sets a value	SEQ	Y	DP
SCA_SQRT	Square-root	SEQ	Y	DP
SCA_SUB	Subtraction	SEQ	Y	DP
SCA_TOVECT	Builds a vector with consecutive scalar values	SEQ	Y	DP
SCA_UTOF	Conversion of unsigned 32-bit integer into float32	SEQ	Y	DP
Matrix processing functions on real				
MAT_ABS	Computes the absolute value of each element	SEQ, ACC	-	DP
MAT_ADD	Addition of two matrix	SEQ, ACC	-	DP
MAT_INV	Computes the inverse of a matrix	SEQ, ACC	-	DP
MAT_SUB	The subtraction of two matrix	SEQ, ACC	-	DP
MAT_MUL	The product of two matrix	SEQ, ACC	-	DP
MAT_OFFSET	Adds an offset to each element of a matrix	SEQ, ACC	y/n	DP
MAT_SCALE	Multiplies each element of a matrix by a scalar	SEQ, ACC	y/n	DP
MAT_TRANS	Transpose of a matrix	SEQ, ACC	-	DP
MAT_VECT2COL	Copies a vector to a matrix column	SEQ, ACC	-	DP
MAT_COL2VECT	Copies a matrix column into a vector	SEQ, ACC	-	DP
Transform processing functions				
RFFT	Real FFT transform	SEQ, ACC	-	DP
FFT	Complex or real FFT and IFFT transform	SEQ, ACC	-	DP
DCT	Discrete cosine transform type 2	SEQ, ACC	-	DP

Table 153. List of processing functions (continued)

Processing function mnemonic	Description	Type	STREAM	BRAM
Conditional processing functions				
IF_ELSE	IF-THEN-ELSE functions	SEQ	-	DP
LOOP	Loop function	SEQ	-	DP
Convolution processing functions				
CNN_CONV2D	2D convolution	ACC	-	SP
CNN_CONVDW	Depth-wise convolution	ACC	-	SP
CNN_CONVPW	Point-wise convolution	ACC	-	SP
CNN_FC	Fully connected layer	ACC	-	SP
CNN_POOL	Pooling layer	ACC	-	SP
Specific processing functions				
COMP	Compares a vector or scalar to two thresholds, and branch according to result.	SEQ	-	DP
MUL_WIN	Element-wise product with a window	SEQ	-	DP
SAT	Performs a saturation to each vector element, using a programmable MAX and MIN values.	SEQ	-	DP
SEND_EVENT	Triggers an event to the SPE.	SEQ	-	-
SET_FLAG	Sets flags to generate interrupt to the CPU.	SEQ	-	-
SET_GPO	Set GPO signals	SEQ	-	-
SET_TRGO	Sets TRGO signals.	SEQ	-	-
SET_BITS	Forces a specific field to a value.	SEQ	Y	DP
CHK_BITS	Checks values of a bitfield.	SEQ	Y	DP
WAIT_COND	Loops until condition is met.	SEQ	Y	DP

20.7 HSP registers

All HSP registers can be read in 8, 16, or 32 bits. All write operations must be performed in 32 bits.

20.7.1 HSP control register (HSP_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This register is used to control several HSP functions such as HSP boot, register access protection, and smart clock gating.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BSTAT[3:0]				Res.	Res.	BARB[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	RSTOK	RSTREQ
r	r	r	r			rw	rw							r	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REGLCK	CDRLCK[1:0]		DCMDEVEN	Res.	Res.	Res.	Res.	SPECGDIS	MMCCGDIS	Res.	CTRLCGDIS	Res.	Res.	Res.	BOOTEN
rs	rs	rs	rw					rw	rw		rw				rs

Bits 31:28 **BSTAT[3:0]**: Boot status

Set and cleared by the HSP. This bitfield shows the status of the boot sequence.
For possible values, refer to *Table 141: BSTAT possible values*.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **BARB[1:0]**: BRAM arbitration

Set and cleared by the CPU.
00: Max arbitration latency is 16 cycles. This setting gives 96% of the BRAM bandwidth to the SPE, when the SPE requests 100% of the BRAM bandwidth. (default after reset).
01: Max arbitration latency is 8 cycles. This setting gives 89% of the BRAM bandwidth to the SPE, when the SPE requests 100% of the BRAM bandwidth.
10: Max arbitration latency is 4 cycles. This setting gives 80% of the BRAM bandwidth to the SPE, when the SPE request 100% of the BRAM bandwidth.
11: Reserved

Note: This bitfield is write-protected when REGLCK = 1.

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **RSTOK**: Reset request accepted

Set by the HSP, cleared by hardware when the HSP is reset. This bit indicates to the CPU that the HSP can be safely reset.
0: The HSP is not yet set into a state which ensures a safe reset. The application must wait until this bit is set (default after reset).
1: The HSP can be safely reset.

Bit 16 **RSTREQ**: Reset request

Set by the CPU, cleared by hardware when the HSP is reset. Writing this bit to 0 has no effect. Read operation returns the effective value.
0: No HSP reset requested (default after reset)
1: HSP reset requested

Bit 15 **REGLCK**: Register write lock access control

Set by the CPU, cleared by hardware when the HSP is reset. Writing this bit to 0 has no effect. Read operation returns the effective value.

0: Registers are accessible by the CPU through the slave AHB interface (default after reset).

1: Selected registers are write-protected and the CPU cannot change the content.

Bits 14:13 **CDRLCK[1:0]**: CRAM and DRAM lock access control

Set by the CPU, cleared by hardware when the HSP is reset. Writing this bit to 0 has no effect. Read operation returns the effective value.

00: CRAM and DRAM are accessible for read and write operations (default after reset).

01: CRAM and DRAM are write protected, but can be read.

1x: CRAM and DRAM are read and write protected.

Bit 12 **DCMDEVEN**: Direct command event enable

Set and cleared by CPU. This bit is used to enable the generation of an event on hsp_txev signal every time the execution of a direct command is completed.

0: Event generation disabled (default after reset)

1: Event generation enabled

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **SPECGDIS**: SPE clock gating disable

Set and cleared by the CPU.

0: SPE clock gating enabled: SPE clocks are locally gated (default after reset).

1: SPE clock gating disabled: SPE clocks are not locally gated.

Bit 6 **MMCCGDIS**: MMC clock gating disable

Set and cleared by the CPU.

0: MMC clock gating enabled: MMC clocks are locally gated (default after reset).

1: MMC clock gating disabled: MMC clocks are not locally gated.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CTRLCGDIS**: HSP_CTRL clock gating disable

Set and cleared by the CPU.

0: HSP_CTRL clock gating enabled: HSP_CTRL clocks are locally gated (default after reset).

1: HSP_CTRL clock gating disabled: HSP_CTRL clocks are not locally gated.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **BOOTEN**: Boot enable control

Set by the CPU and cleared by an HSP reset.

This bit allows the HSP to boot. Writing this bit to 0 has no effect. Read operation returns the effective value.

0: The HSP is not allowed to boot (default after reset).

1: The HSP is allowed to boot, and the SPE executes the code located in the CRAM.

20.7.2 HSP CPU-to-HSP semaphore register (HSP_C2HSEMR)

Address offset: 0x040

Reset value: 0x0000 0000

This register is used by the CPU to synchronize the messages. This register can also be accessed by the SPE processor. When the CPU sets a bit to 1, it sets the corresponding bit. Writing 0 has no effect. Reading returns the effective register value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C2HSEM
															rs

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

Bit 0 **C2HSEM**: CPU to HSP semaphore

Set by the CPU and cleared by the HSP processor. When the CPU writes this bit to 1, it sets the bit. Writing this bit to 0 has no effect. Reading returns the effective bit value.

0: The CPU to HSP MSGB is ready to send a new message (default after reset).

1: The CPU to HSP MSGB contains a message for the HSP not yet read.

20.7.3 HSP CPU-to-HSP message data register (HSP_C2HMSGDR)

Address offset: 0x044

Reset value: 0x0000 0000

This register is used by the CPU to write a message data for the HSP. Writing this register automatically sets C2HSEM in HSP_C2HSEMR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C2HDATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C2HDATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **C2HDATA[31:0]**: CPU message for the HSP

Set and reset by software.

20.7.4 HSP HSP-to-CPU semaphore register (HSP_H2CSEMR)

Address offset: 0x048

Reset value: 0x0000 0000

This register is used by the CPU to synchronize the messages. This register can also be accessed by the HSP processor. When the CPU writes a bit to 1, it clears the corresponding bit. Writing 0 has no effect. Reading returns the effective register value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	H2CSEM
															rc_w1

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

Bit 0 **H2CSEM**: HSP to CPU semaphore

Set by the HSP processor, and cleared by the CPU. When the CPU writes this bit to 1, it clears the bit. Writing this bit to 0 has no effect. Reading returns the effective bit value.
0: The HSP to CPU MSGB does not contain a new message from the HSP (default after reset).
1: The HSP to CPU MSGB contains a new message from the HSP.

20.7.5 HSP HSP-to-CPU message data register (HSP_H2CMSGDR)

Address offset: 0x04C

Reset value: 0x0000 0000

This register is used by the CPU to read a message data sent by the HSP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H2CDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H2CDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **H2CDATA[31:0]**: HSP message for the CPU

Set and reset by software.

20.7.6 HSP DCMD command status register (HSP_DCMDSR)

Address offset: 0x058

Reset value: 0x0000 0000

This register is used by the CPU to check if the current direct command execution is completed.

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

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

Bit 0 **DCBSY**: Direct command busy flag

Set by hardware and reset by SPE. This flag indicates if the SPE is ready to accept a new MSGD.

0: The current direct command execution is completed. A new CMDID can be written in HSP_DCMDIDR (default after reset).

1: The current direct command execution is ongoing.

20.7.7 HSP DCMD pointer status register (HSP_DCMDPTSR)

Address offset: 0x05C

Reset value: 0x0000 0000

This register is used by the CPU to check if the HSP_DCMDPTRx pointer registers have been read by the SPE.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDOG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTRF2	PTRF1	PTRF0
r													r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **RDOG**: Read ongoing indicator

Set and cleared by hardware. This flag indicates if the SPE is currently reading one of the data pointer. The level of this flag can change very fast. It can be used during the debug phase. For example, when the execution of a direct command is abnormally long: if RDOG is stable at 1, it means that the SPE is blocked on the reading of a data pointer. Typically this issue appears when the application did not respect the expected data format for a direct command.

0: SPE does not read a pointer (default after reset).

1: SPE is reading a pointer. The SPE may be blocked if no pointer is provided by the CPU.

Bits 14:3 Reserved, must be kept at reset value.

Bit 2 **PTRF2**: Address pointer 2 flag

Set and cleared by hardware. This flag indicates if the SPE read the last address pointer 2 written by the CPU in HSP_DCMDPTR2.

0: SPE read the address pointer 2. A new address pointer can be written in HSP_DCMDPTR2 (default after reset).

1: The current address pointer 2 has not been read by the SPE.

Bit 1 **PTRF1**: Address pointer 1 flag

Set and cleared by hardware. This flag indicates if the SPE read the last address pointer 1 written by the CPU in HSP_DCMDPTR1.

0: SPE read the address pointer 1. A new address pointer can be written in HSP_DCMDPTR1 (default after reset).

1: The current address pointer 1 has not been read by the SPE.

Bit 0 **PTRF0**: Address pointer 0 flag

Set and cleared by hardware. This flag indicates if the SPE read the last address pointer 0 written by the CPU in HSP_DCMDPTR0.

0: SPE read the address pointer 0. A new address pointer can be written in HSP_DCMDPTR0 (default after reset).

1: The current address pointer 0 has not been read by the SPE.

20.7.8 HSP DCMD command ID register (HSP_DCMDIDR)

Address offset: 0x060

Reset value: 0x0000 0000

This register is used by the CPU to provide the command ID to be executed to the direct command interface (DCMD).

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMDID[15:0]**: Command ID

Set and reset by software. This value represents the command ID of the command to be executed.

20.7.9 HSP DCMD address pointer register x (HSP_DCMDPTRx)

Address offset: 0x064 + 0x4 * x, (x = 0 to 2)

Reset value: 0x0000 0000

This register is used by the CPU to provide the address pointers to the DCMD.

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

Bits 31:0 **PTCPU[31:0]**: Address pointer value

Set and cleared by CPU. The CPU writes in this bitfield the pointer value requested by direct commands. This pointer value represents a peripheral or memory buffer address ‘seen’ by the CPU. This pointer value is automatically translated into a value directly usable by the SPE.

When the CPU writes this register, the corresponding PTRFx flag is set.

20.7.10 HSP event enable register (HSP_EVTENR)

Address offset: 0x080

Reset value: 0x0000 0000

This register is used to control the generation of events, which trigger a task. Write access operations in this register are not allowed when REGLCK is set.

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

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

Bits 22:1 **EVTEN[22:1]**: Events enable

Set and reset by software. EVTEN[n] controls the enabling/disabling of the event number n.

0: Event generation disabled (default after reset)

1: Event generation enabled

Note: This bitfield is write-protected when REGLCK = 1.

Bit 0 Reserved, must be kept at reset value.

20.7.11 HSP event synchronization enable register (HSP_ESYNCENR)

Address offset: 0x084

Reset value: 0x0000 0000

This register is used to enable or disable the synchronization mechanism for the event generation. Write access operations to this register are not allowed when REGLCK is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	CDEGSYNCEN	Res.	Res.	Res.	EVTSYNCEN[22:16]						
					rw				rw	rw	rw	rw	rw	rw	rw

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

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

Bit 26 **CDEGSYNCEN**: CPU dedicated event synchronization enable

Set and cleared by software. This bit controls the enable/disable of the synchronization for the CPU dedicated event.

0: Synchronization disabled (default after reset)

1: Synchronization enabled.

Note: This bit is write-protected when REGLCK = 1.

Bits 25:23 Reserved, must be kept at reset value.

Bits 22:1 **EVTSYNCEN[22:1]**: Event synchronization enable

Set and cleared by software. EVTSYNCEN[n] controls the enable/disable of the synchronization for the event n.

0: Synchronization disabled (default after reset)

1: Synchronization enabled.

Note: This bitfield is write-protected when REGLCK = 1.

Bit 0 Reserved, must be kept at reset value.

20.7.12 HSP event synchronization source register 0 (HSP_ESYNC0SRCR)

Address offset: 0x088

Reset value: 0x0000 0000

This register is used to select the synchronization source for events 1 to 8. Write access operations to this register are not allowed when REGLCK is set, or when the corresponding EVTENx is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EV8SSRC[1:0]		Res.	Res.	EV7SSRC[1:0]		Res.	Res.	EV6SSRC[1:0]		Res.	Res.	EV5SSRC[1:0]	
		rw	rw			rw	rw			rw	rw			rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	EV4SSRC[1:0]		Res.	Res.	EV3SSRC[1:0]		Res.	Res.	EV2SSRC[1:0]		Res.	Res.	EV1SSRC[1:0]	
		rw	rw			rw	rw			rw	rw			rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **EV8SSRC[1:0]**: Event 8 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[8] = 1.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **EV7SSRC[1:0]**: Event 7 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[7] = 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **EV6SSRC[1:0]**: Event 6 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[6] = 1.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **EV5SSRC[1:0]**: Event 5 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[5] = 1.

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

Bits 13:12 **EV4SSRC[1:0]**: Event 4 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[4] = 1.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **EV3SSRC[1:0]**: Event 3 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[8] = 1.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **EV2SSRC[1:0]**: Event 2 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[2] = 1.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **EV1SSRC[1:0]**: Event 1 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[1] = 1.

20.7.13 HSP event synchronization source register 1 (HSP_ESYNC1SRCR)

Address offset: 0x08C

Reset value: 0x0000 0000

This register is used to select the synchronization source for events 9 to 16. Write access operations to this register are not allowed when REGLCK is set, or when the corresponding EVTENx is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EV16SSRC[1:0]		Res.	Res.	EV15SSRC[1:0]		Res.	Res.	EV14SSRC[1:0]		Res.	Res.	EV13SSRC[1:0]	
		rw	rw			rw	rw			rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	EV12SSRC[1:0]		Res.	Res.	EV11SSRC[1:0]		Res.	Res.	EV10SSRC[1:0]		Res.	Res.	EV9SSRC[1:0]	
		rw	rw			rw	rw			rw	rw			rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **EV16SSRC[1:0]**: Event 16 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[16] = 1.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **EV15SSRC[1:0]**: Event 15 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[15] = 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **EV14SSRC[1:0]**: Event 14 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[14] = 1.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **EV13SSRC[1:0]**: Event 13 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[13] = 1.

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

Bits 13:12 **EV12SSRC[1:0]**: Event 12 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[12] = 1.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **EV11SSRC[1:0]**: Event 11 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[11] = 1.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **EV10SSRC[1:0]**: Event 10 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[10] = 1.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **EV9SSRC[1:0]**: Event 9 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[9] = 1.

20.7.14 HSP event synchronization source register 2 (HSP_ESYNC2SRCR)

Address offset: 0x090

Reset value: 0x0000 0000

This register is used to select the synchronization source for events 17 to 22 and 27. Write access operations to this register are not allowed when REGLCK is set or when the corresponding EVTENx = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CDEGSSRC[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EV22SSRC[1:0]		Res.	Res.	EV21SSRC[1:0]	
rw	rw									rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	EV20SSRC[1:0]		Res.	Res.	EV19SSRC[1:0]		Res.	Res.	EV18SSRC[1:0]		Res.	Res.	EV17SSRC[1:0]	
		rw	rw			rw	rw			rw	rw			rw	rw

Bits 31:30 **CDEGSSRC[1:0]**: CPU dedicated event synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when CDEGEN = 1.

Bits 29:22 Reserved, must be kept at reset value.

Bits 21:20 **EV22SSRC[1:0]**: Event 22 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[22] = 1.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **EV21SSRC[1:0]**: Event 21 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[21] = 1.

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

Bits 13:12 **EV20SSRC[1:0]**: Event 20 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[20] = 1.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **EV19SSRC[1:0]**: Event 19 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[19] = 1.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **EV18SSRC[1:0]**: Event 18 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[18] = 1.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **EV17SSRC[1:0]**: Event 17 synchronization source
Set and cleared by software.
00: **tsk_cmp**[0] selected as synchronization source
01: **tsk_cmp**[1] selected as synchronization source
10: **tsk_cmp**[2] selected as synchronization source
11: **tsk_cmp**[3] selected as synchronization source
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[17] = 1.

20.7.15 HSP pending event level register (HSP_PEVTLR)

Address offset: 0x098

Reset value: 0x0000 0000

This register shows event states before the priority encoder. This register is mainly dedicated to debugging purposes.

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

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:0 **EVTL[27:0]**: Event level
Set and cleared by hardware. This bitfield shows the current events level. Bit n correspond to event n.
0: Level low (inactive)
1: Level high (pending)

20.7.16 HSP interface enable register (HSP_ITFENR)

Address offset: 0x0A0

Reset value: 0x0000 0000

This register is used by the application to enable the STREAM and TRGITF peripheral interfaces. CPU write access operations to this register are not allowed when REGLCK is set. This register can also be modified by the SPE.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HDEGEN	HSEGEN	CDEGEN	CSEGEN	DCMDDIS	Res.	TRGI9EN	TRGI8EN	TRGI7EN	TRGI6EN	TRGI5EN	TRGI4EN	TRGI3EN	TRGI2EN	TRGI1EN	TRGI0EN
r	r	rw	rw	rwo		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TRGOEN	Res.	Res.	Res.	STREAMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw				rw								

Bit 31 **HDEGEN**: HDEG interface enable bit
Set and cleared by software.
0: HDEG interface disabled, and maintained in reset
1: HDEG interface enabled

Bit 30 **HSEGEN**: HSEG interface enable bit
Set and cleared by software.
0: HSEG interface disabled, and maintained in reset
1: HSEG interface enabled

Bit 29 **CDEGEN**: CDEG interface enable bit
Set and cleared by software.
0: CDEG interface disabled, and maintained in reset
1: CDEG interface enabled
Note: This bit is write-protected when REGLCK = 1.

Bit 28 **CSEGEN**: CSEG interface enable bit
Set and cleared by software.
0: CSEG interface disabled, and maintained in reset
1: CSEG interface is enabled
Note: This bit is write-protected when REGLCK = 1.

Bit 27 **DCMDDIS**: Direct command interface disable bit
Set and cleared by software. This bit can only be written once, after a HSP reset. By default the direct command interface is enabled. Reading returns the effective value.
0: Writing this bit to 0 enables the DCMD interface, and prevents other applications to disable it (default after reset).
1: Writing this bit to 1 disables the DCMD interface. Only an HSP reset can re enable it.
Note: This bit is write-protected when REGLCK = 1.

Bit 26 Reserved, must be kept at reset value.

Bits 25:16 **TRGIxEN**: TRGINx enable bit (x = 9 to 0)
Set and cleared by software.
0: TRGINx disabled, all the TRGIN logic maintained in reset
1: TRGINx enabled
Note: This bit is write-protected when REGLCK = 1.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **TRGOEN**: TRGO enable bit
Set and cleared by software.
0: TRGO function disabled. It is not possible to generate trigger pulses on hsp_trgo[3:0], nor to change the level of hsp_gpo[7:0].
1: TRGO function enabled
Note: This bit is write-protected when REGLCK = 1.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **STREAMEN**: STREAM enable
Set and cleared by software.
0: STREAM disabled
1: STREAM enabled
Note: This bit is write-protected when REGLCK = 1.

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

20.7.17 HSP event source register 0 (HSP_EVTSRC0R)

Address offset: 0x0B4

Reset value: 0x0000 0000

This register is used to select the event source for the priority encoder inputs 1 to 8. Write access operations to this register are not allowed when REGLCK is set, or when the corresponding EVTENx is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	EVT8SRC[2:0]			Res.	EVT7SRC[2:0]			Res.	EVT6SRC[2:0]			Res.	EVT5SRC[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EVT4SRC[2:0]			Res.	EVT3SRC[2:0]			Res.	EVT2SRC[2:0]			Res.	EVT1SRC[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **EVT8SRC[2:0]**: Event source selection for priority encoder input 8
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF3 is selected (**buff_evt**[3])
011: Event provided by the TRGIN0 is selected (**trgin_evt**[0])
100: Event provided by the TRGIN5 is selected (**trgin_evt**[5])
101: Event provided by the HSEG is selected (**hseg_evt**[8])
110: Event provided by the CSEG is selected (**cseg_evt**[8])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[8] = 1.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **EVT7SRC[2:0]**: Event source selection for priority encoder input 7
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF2 is selected (**buff_evt**[2])
011: Event provided by the TRGIN4 is selected (**trgin_evt**[4])
100: Event provided by the TRGIN9 is selected (**trgin_evt**[9])
101: Event provided by the HSEG is selected (**hseg_evt**[7])
110: Event provided by the CSEG is selected (**cseg_evt**[7])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[7] = 1.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **EVT6SRC[2:0]**: Event source selection for priority encoder input 6
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF1 is selected (**buff_evt**[1])
011: Event provided by the TRGIN3 is selected (**trgin_evt**[3])
100: Event provided by the TRGIN8 is selected (**trgin_evt**[8])
101: Event provided by the HSEG is selected (**hseg_evt**[6])
110: Event provided by the CSEG is selected (**cseg_evt**[6])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[6] = 1.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **EVT5SRC[2:0]**: Event source selection for priority encoder input 5
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF0 is selected (**buff_evt**[0])
011: Event provided by the TRGIN2 is selected (**trgin_evt**[2])
100: Event provided by the TRGIN7 is selected (**trgin_evt**[7])
101: Event provided by the HSEG is selected (**hseg_evt**[5])
110: Event provided by the CSEG is selected (**cseg_evt**[5])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[5] = 1.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **EVT4SRC[2:0]**: Event source selection for priority encoder input 4
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF3 is selected (**buff_evt**[3])
011: Event provided by the TRGIN1 is selected (**trgin_evt**[1])
100: Event provided by the TRGIN6 is selected (**trgin_evt**[6])
101: Event provided by the HSEG is selected (**hseg_evt**[4])
110: Event provided by the CSEG is selected (**cseg_evt**[4])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[4] = 1.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **EVT3SRC[2:0]**: Event source selection for priority encoder input 3
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF2 is selected (**buff_evt**[2])
011: Event provided by the TRGIN0 is selected (**trgin_evt**[0])
100: Event provided by the TRGIN5 is selected (**trgin_evt**[5])
101: Event provided by the HSEG is selected (**hseg_evt**[3])
110: Event provided by the CSEG is selected (**cseg_evt**[3])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[3] = 1.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **EVT2SRC[2:0]**: Event source selection for priority encoder input 2
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF1 is selected (**buff_evt**[1])
011: Event provided by the TRGIN4 is selected (**trgin_evt**[4])
100: Event provided by the TRGIN9 is selected (**trgin_evt**[9])
101: Event provided by the HSEG is selected (**hseg_evt**[2])
110: Event provided by the CSEG is selected (**cseg_evt**[2])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[2] = 1.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **EVT1SRC[2:0]**: Event source selection for priority encoder input 1
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF0 is selected (**buff_evt**[0])
011: Event provided by the TRGIN3 is selected (**trgin_evt**[3])
100: Event provided by the TRGIN8 is selected (**trgin_evt**[8])
101: Event provided by the HSEG is selected (**hseg_evt**[1])
110: Event provided by the CSEG is selected (**cseg_evt**[1])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[1] = 1.

20.7.18 HSP event source register 1 (HSP_EVTSRC1R)

Address offset: 0x0B8

Reset value: 0x0000 0000

This register is used to select the event source for the priority encoder inputs 9 to 16. Write access operations to this register are not allowed when REGLCK is set or when the corresponding EVTENx is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	EVT16SRC[2:0]			Res.	EVT15SRC[2:0]			Res.	EVT14SRC[2:0]			Res.	EVT13SRC[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EVT12SRC[2:0]			Res.	EVT11SRC[2:0]			Res.	EVT10SRC[2:0]			Res.	EVT9SRC[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **EVT16SRC[2:0]**: Event source selection for priority encoder input 16
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF3 is selected (**buff_evt**[3])
011: Event provided by the TRGIN3 is selected (**trgin_evt**[3])
100: Event provided by the TRGIN8 is selected (**trgin_evt**[8])
101: Event provided by the HSEG is selected (**hseg_evt**[16])
110: Event provided by the CSEG is selected (**cseg_evt**[16])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[16] = 1.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **EVT15SRC[2:0]**: Event source selection for priority encoder input 15
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF2 is selected (**buff_evt**[2])
011: Event provided by the TRGIN2 is selected (**trgin_evt**[2])
100: Event provided by the TRGIN7 is selected (**trgin_evt**[7])
101: Event provided by the HSEG is selected (**hseg_evt**[15])
110: Event provided by the CSEG is selected (**cseg_evt**[15])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[15] = 1.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **EVT14SRC[2:0]**: Event source selection for priority encoder input 14
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF1 is selected (**buff_evt**[1])
011: Event provided by the TRGIN1 is selected (**trgin_evt**[1])
100: Event provided by the TRGIN7 is selected (**trgin_evt**[6])
101: Event provided by the HSEG is selected (**hseg_evt**[14])
110: Event provided by the CSEG is selected (**cseg_evt**[14])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[14] = 1.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **EVT13SRC[2:0]**: Event source selection for priority encoder input 13
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF0 is selected (**buff_evt**[0])
011: Event provided by the TRGIN0 is selected (**trgin_evt**[0])
100: Event provided by the TRGIN5 is selected (**trgin_evt**[5])
101: Event provided by the HSEG is selected (**hseg_evt**[13])
110: Event provided by the CSEG is selected (**cseg_evt**[13])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[13] = 1.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **EVT12SRC[2:0]**: Event source selection for priority encoder input 12
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF3 is selected (**buff_evt**[3])
011: Event provided by the TRGIN4 is selected (**trgin_evt**[4])
100: Event provided by the TRGIN9 is selected (**trgin_evt**[9])
101: Event provided by the HSEG is selected (**hseg_evt**[12])
110: Event provided by the CSEG is selected (**cseg_evt**[12])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[12] = 1.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **EVT11SRC[2:0]**: Event source selection for priority encoder input 11
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF2 is selected (**buff_evt**[2])
011: Event provided by the TRGIN3 is selected (**trgin_evt**[3])
100: Event provided by the TRGIN8 is selected (**trgin_evt**[8])
101: Event provided by the HSEG is selected (**hseg_evt**[11])
110: Event provided by the CSEG is selected (**cseg_evt**[11])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[11] = 1.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **EVT10SRC[2:0]**: Event source selection for priority encoder input 10
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF1 is selected (**buff_evt**[1])
011: Event provided by the TRGIN2 is selected (**trgin_evt**[2])
100: Event provided by the TRGIN7 is selected (**trgin_evt**[7])
101: Event provided by the HSEG is selected (**hseg_evt**[10])
110: Event provided by the CSEG is selected (**cseg_evt**[10])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[10] = 1.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **EVT9SRC[2:0]**: Event source selection for priority encoder input 9
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF0 is selected (**buff_evt**[0])
011: Event provided by the TRGIN1 is selected (**trgin_evt**[1])
100: Event provided by the TRGIN6 is selected (**trgin_evt**[6])
101: Event provided by the HSEG is selected (**hseg_evt**[9])
110: Event provided by the CSEG is selected (**cseg_evt**[9])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[9] = 1.

20.7.19 HSP event source register 2 (HSP_EVTSRC2R)

Address offset: 0x0BC

Reset value: 0x0000 0000

This register is used to select the event source for the priority encoder inputs 17 to 22. Write access operations to this register are not allowed when REGLCK is set, or when the corresponding EVTENx is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVT22SRC[2:0]			Res.	EVT21SRC[2:0]		
									rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EVT20SRC[2:0]			Res.	EVT19SRC[2:0]			Res.	EVT18SRC[2:0]			Res.	EVT17SRC[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

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

Bits 22:20 **EVT22SRC[2:0]**: Event source selection for priority encoder input 22
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF1 is selected (**buff_evt**[1])
011: Event provided by the TRGIN4 is selected (**trgin_evt**[4])
100: Event provided by the TRGIN9 is selected (**trgin_evt**[9])
101: Event provided by the HSEG is selected (**hseg_evt**[22])
110: Event provided by the CSEG is selected (**cseg_evt**[22])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[22] = 1.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **EVT21SRC[2:0]**: Event source selection for priority encoder input 21
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF0 is selected (**buff_evt**[0])
011: Event provided by the TRGIN3 is selected (**trgin_evt**[3])
100: Event provided by the TRGIN8 is selected (**trgin_evt**[8])
101: Event provided by the HSEG is selected (**hseg_evt**[21])
110: Event provided by the CSEG is selected (**cseg_evt**[21])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[21] = 1.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **EVT20SRC[2:0]**: Event source selection for priority encoder input 20
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF3 is selected (**buff_evt**[3])
011: Event provided by the TRGIN2 is selected (**trgin_evt**[2])
100: Event provided by the TRGIN7 is selected (**trgin_evt**[7])
101: Event provided by the HSEG is selected (**hseg_evt**[20])
110: Event provided by the CSEG is selected (**cseg_evt**[20])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[20] = 1.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **EVT19SRC[2:0]**: Event source selection for priority encoder input 19
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF2 is selected (**buff_evt**[2])
011: Event provided by the TRGIN1 is selected (**trgin_evt**[1])
100: Event provided by the TRGIN6 is selected (**trgin_evt**[6])
101: Event provided by the HSEG is selected (**hseg_evt**[19])
110: Event provided by the CSEG is selected (**cseg_evt**[19])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[19] = 1.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **EVT18SRC[2:0]**: Event source selection for priority encoder input 18
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF1 is selected (**buff_evt**[1])
011: Event provided by the TRGIN0 is selected (**trgin_evt**[0])
100: Event provided by the TRGIN5 is selected (**trgin_evt**[5])
101: Event provided by the HSEG is selected (**hseg_evt**[18])
110: Event provided by the CSEG is selected (**cseg_evt**[18])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[18] = 1.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **EVT17SRC[2:0]**: Event source selection for priority encoder input 17
Set and cleared by software.
000: Reserved, no event selected
001: Reserved, no event selected
010: Event provided by the BUFF0 is selected (**buff_evt**[0])
011: Event provided by the TRGIN4 is selected (**trgin_evt**[4])
100: Event provided by the TRGIN9 is selected (**trgin_evt**[9])
101: Event provided by the HSEG is selected (**hseg_evt**[17])
110: Event provided by the CSEG is selected (**cseg_evt**[17])
Other: same as position 101
Note: This bitfield is write-protected when REGLCK = 1 or when EVTEN[17] = 1.

20.7.20 HSP BUFF configuration register (HSP_BUFFCFGR)

Address offset: 0x0F4

Reset value: 0x0000 0000

This register is used by the application to select the direction of each buffer (C2H or H2C), and to configure the buffer combiner. Write access operations to this register are not allowed when REGLCK is set or when BUFFEN is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMB 2	COMB 1	COMB 0
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2FEN	Res.	Res.	Res.	Res.	BUFF3 DIR	BUFF2 DIR	BUFF1 DIR	BUFF0 DIR
							rw					rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **COMBx**: BUFCMB control for **buff_evt**[x] (x = 2 to 0)

Set and cleared by software. This bit defines how **buff_evt**[x] is activated.

0: **buff_evt**[x] activated when **buff_att**[x] is activated

1: **buff_evt**[x] activated when **buff_att**[x] and **buff_evt**[x+1] are activated

Note: This bit is write-protected when REGLCK = 1 or when BUFFEN = 1.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **I2FEN**: Integer to float32 conversion

Set and cleared by software. The integer to float only works for the RX buffers. The integer number must have one of the following format:

- signed integers
- unsigned 8-, 16-, or 24-bit integers

0: Integer to float32 conversion disabled

1: Integer to float32 conversion enabled

Note: This bit is write-protected when REGLCK = 1 or when BUFFEN = 1.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **BUFFxDIR**: Direction selection of BUFFx (x = 3 to 0)

Set and cleared by software.

0: BUFFx direction is C2H (means written by the CPU and read by the HSP).

1: BUFFx direction is H2C (means read by the CPU and written by the HSP).

Note: This bit is write-protected when REGLCK = 1 or when BUFFEN = 1.

20.7.21 HSP BUFFx data register x (HSP_BUFFxDR)

Address offset: 0x0F8 + 0x4 * x, (x = 0 to 3)

Reset value: 0x0000 0000

This register is used to read or write data into a C2H or H2CBUFF.

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

Bits 31:0 **BUFFDAT[31:0]**: Data buffer

Set and cleared by software. The application can read H2CBUFFs, and write C2HBUFFs. Writing into a H2CBUFF has no effect, reading a C2HBUFF returns an invalid data.

20.7.22 HSP TRGIN configuration register (HSP_TRGINCFGR)

Address offset: 0x120

Reset value: 0x0000 0000

This register is used to enable and select the polarity of each TRGIN input. Write access operations to this register are not allowed when REGLCK is set or when TRGIxEN is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TRG9POL	TRG8POL	TRG7POL	TRG6POL	TRG5POL	TRG4POL	TRG3POL	TRG2POL	TRG1POL	TRG0POL
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 9:0 **TRGxPOL**: Polarity selection for TRGINx (x = 9 to 0)

Set and cleared by software.

0: TRGINx triggers on rising edge of the input signal.

1: TRGINx triggers on falling edge of the input signal.

Note: This bit is write-protected when REGLCK = 1 or when TRGI[x]EN= 1.

20.7.23 HSP TRGO configuration register (HSP_TRGOCFGR)

Address offset: 0x124

Reset value: 0x0000 0000

This register is used to enable and select the signal sources generating a trigger output. Write access operations to this register are not allowed when REGLCK is set or when TRGOEN is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGO3SRC[1:0]		TRGO2SRC[1:0]		TRGO1SRC[1:0]		TRGO0SRC[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **TRGOxSRC[1:0]**: Trigger source selection for **hsp_trgo**[x] (x = 3 to 0)
Set and cleared by software.
00: **hsp_trgo**[x] disabled
01: **buff_trgo**[x] selected as trigger source
10: **spe_trgo**[x] selected as trigger source
11: **tsc_trgo** selected as trigger source
Note: This bitfield is write-protected when REGLCK = 1 or when TRGOEN = 1.

20.7.24 HSP TRGIN input selection register 0 (HSP_TRGINSELR0)

Address offset: 0x150

Reset value: 0x0000 0000

This register is used to select the input trigger signal for TRGIN0 to TRGIN3. Write access operations to this register are not allowed when REGLCK is set or when the corresponding TRGIxEN is set.

Note: The number of valid positions depends on the number of trigger inputs implemented in the product (see Section 20.3: HSP implementation).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TRG3SEL[5:0]						Res.	Res.	TRG2SEL[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	TRG1SEL[5:0]						Res.	Res.	TRG0SEL[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **TRG3SEL[5:0]**: Input trigger selection for TRGIN3
Set and cleared by software.
0: **hsp_trgi**[0] input signal selected
1: **hsp_trgi**[1] input signal selected
x: **hsp_trgi**[x] input signal selected (x must be lower than 20)
Note: This bit is write-protected when REGLCK = 1 or when TRGI3EN= 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **TRG2SEL[5:0]**: Input trigger selection for TRGIN2
Set and cleared by software.
0: **hsp_trgi**[0] input signal selected
1: **hsp_trgi**[1] input signal selected
x: **hsp_trgi**[x] input signal selected (x must be lower than 20)
Note: This bit is write-protected when REGLCK = 1 or when TRGI2EN= 1.

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

Bits 13:8 **TRG1SEL[5:0]**: Input trigger selection for TRGIN1
Set and cleared by software.
0: **hsp_trgi**[0] input signal selected
1: **hsp_trgi**[1] input signal selected
x: **hsp_trgi**[x] input signal selected (x must be lower than 20)
Note: This bit is write-protected when REGLCK = 1 or when TRGI1EN= 1.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **TRG0SEL[5:0]**: Input trigger selection for TRGIN0
Set and cleared by software.
0: **hsp_trgi**[0] input signal selected
1: **hsp_trgi**[1] input signal selected
x: **hsp_trgi**[x] input signal selected (x must be lower than 20)
Note: This bit is write-protected when REGLCK = 1 or when TRGI0EN= 1.

20.7.25 HSP TRGIN input selection register 1 (HSP_TRGINSELR1)

Address offset: 0x154

Reset value: 0x0000 0000

This register is used to select the input trigger signal for TRGIN4 to TRGIN7. Write access operations to this register are not allowed when REGLCK is set or when the corresponding TRGIxEN is set.

Note: The number of valid positions depends on the number of trigger inputs implemented in the product (see Section 20.3: HSP implementation).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TRG7SEL[5:0]						Res.	Res.	TRG6SEL[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	TRG5SEL[5:0]						Res.	Res.	TRG4SEL[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **TRG7SEL[5:0]**: Input trigger selection for TRGIN7
Set and cleared by software.
0: **hsp_trgi**[0] input signal selected
1: **hsp_trgi**[1] input signal selected
x: **hsp_trgi**[x] input signal selected (x must be lower than 20)
Note: This bit is write-protected when REGLCK = 1 or when TRGI7EN= 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **TRG6SEL[5:0]**: Input trigger selection for TRGIN6
Set and cleared by software.
0: **hsp_trgi**[0] input signal selected
1: **hsp_trgi**[1] input signal selected
x: **hsp_trgi**[x] input signal selected (x must be lower than 20)
Note: This bit is write-protected when REGLCK = 1 or when TRGI6EN= 1.

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

Bits 13:8 **TRG5SEL[5:0]**: Input trigger selection for TRGIN5
Set and cleared by software.
0: **hsp_trgi**[0] input signal selected
1: **hsp_trgi**[1] input signal selected
x: **hsp_trgi**[x] input signal selected (x must be lower than 20)
Note: This bit is write-protected when REGLCK = 1 or when TRGI5EN= 1.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **TRG4SEL[5:0]**: Input trigger selection for TRGIN4
Set and cleared by software.
0: **hsp_trgi**[0] input signal selected
1: **hsp_trgi**[1] input signal selected
x: **hsp_trgi**[x] input signal selected (x must be lower than 20)
Note: This bit is write-protected when REGLCK = 1 or when TRGI4EN= 1.

20.7.26 HSP TRGIN input selection register 2 (HSP_TRGINSELR2)

Address offset: 0x158

Reset value: 0x0000 0000

This register is used to select the input trigger signal for TRGIN8 to TRGIN9. Write access operations to this register are not allowed when REGLCK is set or when the corresponding TRGIxEN is set.

Note: The number of valid positions depends on the number of trigger inputs implemented in the product (see Section 20.3: HSP implementation).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	TRG9SEL[5:0]						Res.	Res.	TRG8SEL[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **TRG9SEL[5:0]**: Input trigger selection for TRGIN9
Set and cleared by software.
0: **hsp_trgi**[0] input signal selected
1: **hsp_trgi**[1] input signal selected
x: **hsp_trgi**[x] input signal selected (x must be lower than 20)
Note: This bit is write-protected when REGLCK = 1 or when TRGI9EN= 1.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **TRG8SEL[5:0]**: Input trigger selection for TRGIN8
Set and cleared by software.
0: **hsp_trgi**[0] input signal selected
1: **hsp_trgi**[1] input signal selected
x: **hsp_trgi**[x] input signal selected (x must be lower than 20)
Note: This bit is write-protected when REGLCK = 1 or when TRGI8EN= 1.

20.7.27 HSP CPU shared event generator register (HSP_CSEGR)

Address offset: 0x12C

Reset value: 0x0000 0000

This register is used by the application software to generate an event. Writing 1 sets the corresponding event. Writing 0 has no effect. Reading shows the events status.

Note: If, at the same time, the software sets one of the CSEVT[22:1] bits, and the hardware clears the same bit, the corresponding bit is cleared.

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

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

Bits 22:1 **CSEVT[22:1]**: CPU shared software event
Set by software, cleared by hardware. The application can activated any of the shared event inputs (**cseg_evt**[22:1]) (see *Section 20.3: HSP implementation*).
0: Writing 0 has no effect. Reading 0 means that the corresponding event trigger is not waiting to be served.
1: Writing 1 activates the corresponding event. Reading 1 means that the corresponding event trigger is waiting to be served.

Bit 0 Reserved, must be kept at reset value.

20.7.28 HSP CPU dedicated event generation register (HSP_CDEGR)

Address offset: 0x130

Reset value: 0x0000 0000

This register is used by the application to request the execution of a specific process. Writing into this register activates the event 27.

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

Bit 31 **CDEGBSY**: CPU dedicated event generator busy

Set by hardware and cleared by SPE when the event is taken into account. This flag informs the CPU if the CDEG is ready to accept the generation of a new event.

0: The CDEG ready

1: The CDEG busy: previous event still under processing

Bits 30:6 Reserved, must be kept at reset value.

Bits 5:0 **CTSKN[5:0]**: CPU task number

Set and cleared by software. This bitfield defines the task number associated to **prc_evt**[27]. When the priority encoder selected prc_evt[27], the task given by CTSKN[5:0] is executed.

20.7.29 HSP task comparator unit register (HSP_TCUCFGR)

Address offset: 0x340

Reset value: 0x0000 0000

This register is used to configure the task comparators 0 to 3 of the TCU. Write access operations to this register are not allowed when REGLCK is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TC3EN	Res.	TSKCMP3[5:0]						TC2EN	Res.	TSKCMP2[5:0]					
rw		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TC1EN	Res.	TSKCMP1[5:0]						TC0EN	Res.	TSKCMP0[5:0]					
rw		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bit 31 **TC3EN**: Task comparator 3 enable

Set and cleared by software.

0: TSKCMP3 output disabled

1: TSKCMP3 output enabled

Note: This bit is write-protected when REGLCK = 1.

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **TSKCMP3[5:0]**: Task compare 3

Set and cleared by software to define which task must be detected by TSKCMP3 output. It is recommended to change this value when TC3EN = 0. This bitfield must be lower or equal to 39.

Note: This bitfield is write-protected when REGLCK = 1.

Bit 23 **TC2EN**: Task comparator 2 enable

Set and cleared by software.
0: TSKCMP2 output disabled
1: TSKCMP2 output enabled

Note: This bit is write-protected when REGLCK = 1.

Bit 22 Reserved, must be kept at reset value.

Bits 21:16 **TSKCMP2[5:0]**: Task compare 2

Set and cleared by software to define which task must be detected by TSKCMP2 output. It is recommended to change this value when TC2EN = 0. This bitfield must be lower or equal to 39.

Note: This bitfield is write-protected when REGLCK = 1.

Bit 15 **TC1EN**: Task comparator 1 enable

Set and cleared by software.
0: TSKCMP1 output disabled
1: TSKCMP1 output enabled

Note: This bit is write-protected when REGLCK = 1.

Bit 14 Reserved, must be kept at reset value.

Bits 13:8 **TSKCMP1[5:0]**: Task compare 1

Set and cleared by software to define which task must be detected by TSKCMP1 output. It is recommended to change this value when TC1EN = 0. This bitfield must be lower or equal to 39.

Note: This bitfield is write-protected when REGLCK = 1.

Bit 7 **TC0EN**: Task comparator 0 enable

Set and cleared by software.
0: TSKCMP0 output disabled
1: TSKCMP0 output enabled

Note: This bit is write-protected when REGLCK = 1.

Bit 6 Reserved, must be kept at reset value.

Bits 5:0 **TSKCMP0[5:0]**: Task compare 0

Set and cleared by software to define which task must be detected by TSKCMP0 output. It is recommended to change this value when TC0EN = 0. This bitfield must be lower or equal to 39.

Note: This bitfield is write-protected when REGLCK = 1.

20.7.30 HSP task overlap control register (HSP_TOVLPCR)

Address offset: 0x344

Reset value: 0x0000 0000

This register is used to configure the task overlap function of the TCU. Write access operations to this register are not allowed when REGLCK is set.

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

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

Bits 21:16 **HTNB[5:0]**: Higher task number value

The selected task group where the task overlap is checked, is strictly higher than LTNB, and lower than HTNB[5:0]. This bitfield has no effect if TOVLPEN = 0.

Note: This bitfield is write-protected when REGLCK = 1.

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

Bits 13:8 **LTNB[5:0]**: Lower task number value

The selected task group where the task overlap is checked, is strictly higher than LTNB, and lower than HTNB[5:0]. This bitfield has no effect if TOVLPEN = 0.

Note: This bitfield is write-protected when REGLCK = 1.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **TOVLPEN**: Task overlap enable

Set and cleared by software.

0: Task overlap detector disabled

1: Task overlap detector enabled

Note: This bit is write-protected when REGLCK = 1.

20.7.31 HSP SNOOP register (HSP_SNPR)

Address offset: 0x378

Reset value: 0x0000 0000

This register is used by the CPU to select the bank of signals to observe (see *Section 20.4.18: Snoop function (SNOOP)*).

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

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **SNPSELB[2:0]**: SNOOP group B signal selection
Set and cleared by software.
000: HSP_SNP[5:3] outputs are disabled.
001: Selects signal set 1 to HSP_SNP[5:3] outputs.
010: Selects signal set 2 to HSP_SNP[5:3] outputs.
011: Selects signal set 3 to HSP_SNP[5:3] outputs.
100: Selects signal set 4 to HSP_SNP[5:3] outputs.
101: Selects signal set 5 to HSP_SNP[5:3] outputs.
110: Selects signal set 6 to HSP_SNP[5:3] outputs.
111: Selects signal set 7 to HSP_SNP[5:3] outputs.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SNPSELA[2:0]**: SNOOP group A signal selection
Set and cleared by software.
000: HSP_SNP[2:0] outputs are disabled.
001: Selects signal set 1 to HSP_SNP[2:0] outputs.
010: Selects signal set 2 to HSP_SNP[2:0] outputs.
011: Selects signal set 3 to HSP_SNP[2:0] outputs.
100: Selects signal set 4 to HSP_SNP[2:0] outputs.
101: Selects signal set 5 to HSP_SNP[2:0] outputs.
110: Selects signal set 6 to HSP_SNP[2:0] outputs.
111: Selects signal set 7 to HSP_SNP[2:0] outputs.

20.7.32 HSP conflict counter register (HSP_CCNTR)

Address offset: 0x37C

Reset value: 0x0000 0000

This register is used by the CPU to get the number of conflicting accesses detected in the BRAM memory. A conflict does not mean that an error occurred. It means that several masters attempted to access the BRAM, and some of these accesses have been delayed by the local arbiter.

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

Bits 31:0 **CCNTR[31:0]**: Conflict counter
Set by hardware. This bitfield gives the amount of conflicts. This counter is reset by an HSP reset or when the CCNTREN bit is cleared.

20.7.33 HSP data capture register (HSP_CAPDR)

Address offset: 0x380

Reset value: 0x0000 0000

This register is used by the CPU to read the timestamp of the last captured task number change. This register can take two different formats according to FRCNTEN. Here is the format when FRCNTEN = 1.

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

Bits 31:26 **TSKNB[5:0]**: Last captured task number
Set by hardware. This bitfield gives the last captured task number.

Bits 25:22 **IFCNT[3:0]**: IF instruction counter
Set by hardware. This bitfield gives the number of IF instructions executed since the previous processing function.

Bits 21:14 **IFHIST[7:0]**: IF condition history
Set by hardware.
IFHIST[x] = 1 means that the comparison performed on the x^{th} IF instruction after the execution of the previous processing function was true.
IFHIST[x] = 0 means that the comparison was false.

Bits 13:0 **TSTAMP[13:0]**: Time-stamp value
Set by hardware. This bitfield gives the time-stamp of the last captured event.

20.7.34 HSP data capture register [alternate] (HSP_CAPDR)

Address offset: 0x380

Reset value: 0x0000 0000

This register is used by the CPU to read the timestamp of the last captured task number change. This register can take two different formats according to FRCNTEN. Here is the format when FRCNTEN = 0.

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

Bits 31:26 **TSKNB[5:0]**: Last captured task number
Set by hardware. This bitfield gives the last captured task number.

Bits 25:22 **IFCNT[3:0]**: IF instruction counter
Set by hardware. This bitfield gives the number of IF instructions executed since the previous processing function.

Bits 21:16 **PFCTNB[5:0]**: Last captured processing function number
Set by hardware. This bitfield gives the last captured processing function. It is forced to 0 when CAPMOD = 2.

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **IFHIST[14:0]**: IF condition history
Set by hardware.
IFHIST[x] = 1 means that the comparison performed on the x^{th} IF instruction after the execution of the previous processing function was true.
IFHIST[x] = 0 means that the comparison was false.

20.7.35 HSP capture control register (HSP_CAPCR)

Address offset: 0x384

Reset value: 0x0000 0000

This register is used to control the capture function.

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

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

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **CCNTREN**: Conflict counter enable
Set and cleared by software. The conflict counter can be used to count the amount of conflicts reported by the MMC to access to the BRAM.
0: Conflict counter disabled, and reset to 0
1: Conflict counter enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **TSKFLT**: Task filter enable
Set and cleared by software.
0: The trace controller asserts the **tsc_pfcttd** signal every time the processing function is executed, independently to the task number running.
1: The trace controller asserts the **tsc_pfcttd** signal every time the processing function executed when the task selected by TSKCMP0[5:0] is running.

Bits 11:9 Reserved, must be kept at reset value.

Bits 8:4 **PRESC[4:0]**: Time-stamp prescaler value
Set and cleared by software. This bitfield gives the division ratio to define the time-stamp granularity.
00000: Clock division is 1 (bypassed).
00001: Clock division is hsp_hclk_core_ck / 2.
00010: Clock division is hsp_hclk_core_ck / 4.
00011: Clock division is hsp_hclk_core_ck / 8.
00100: Clock division is hsp_hclk_core_ck / 16.
00101: Clock division is hsp_hclk_core_ck / 32.
00110: Clock division is hsp_hclk_core_ck / 64.
00111: Clock division is hsp_hclk_core_ck / 128.
01000: Clock division is hsp_hclk_core_ck / 256.
01001: Clock division is hsp_hclk_core_ck / 512.
01010: Clock division is hsp_hclk_core_ck / 1024.
01011: Clock division is hsp_hclk_core_ck / 2048.
01100: Clock division is hsp_hclk_core_ck / 4096.
01101: Clock division is hsp_hclk_core_ck / 8192.
01110: Clock division is hsp_hclk_core_ck / 16384.
01111: Clock division is hsp_hclk_core_ck / 32768.
Other: Reserved

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FRCNTEN**: Free-running counter enable
Set and cleared by software.
0: The free-running counter and prescaler are disabled and reset.
1: The free-running counter and prescaler are enabled.

Bits 1:0 **CAPMOD[1:0]**: Capture mode
Set and cleared by software.
00: Capture function is disabled.
01: A capture is performed every time the SPE breaks.
10: A capture is performed every time the task number changes.
11: A capture is performed every time the processing function number changes.

20.7.36 HSP break output configuration register x (HSP_BKOxCFGR)

Address offset: 0x168 + 0x4 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register is used by the application to configure the break outputs 0, 1, 2 or 3 (**hsp_break_out**[3:0] signals). Write access operations to this register are not allowed when REGLCK is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	HDEGOVEN	FWEREN	Res.	Res.	Res.	Res.	PFCT31EN	PFCT30EN	PFCT29EN	PFCT28EN	Res.	OPCEREN	FPUEREN	ACCEREN
		rw	rw					rw	rw	rw	rw		rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **HDEGOVEN**: HSP dedicated event generator overrun break enable for HDEGOVRF
Set and cleared by software.
0: HDEGOVRF flag does not generate a break.
1: HDEGOVRF flag generates a break when asserted.
Note: This bit is write-protected when REGLCK = 1.

Bit 12 **FWEREN**: Firmware error break enable for FWERRF
Set and cleared by software.
0: FWERRF flag does not generate a break.
1: FWERRF flag generates a break when asserted.
Note: This bit is write-protected when REGLCK = 1.

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:4 **PFCTxEN**: Processing function flag break enable for PFCTF[x] (x = 31 to 28)
Set and cleared by software.
0: PFCTF[x] does not generate a break.
1: PFCTF[x] generates a break when asserted.
Note: This bit is write-protected when REGLCK = 1.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **OPCEREN**: Opcode error break enable for OPCOERRF
Set and cleared by software.
0: OPCOERRF flag does not generate a break.
1: OPCOERRF flag generates a break when asserted.
Note: This bit is write-protected when REGLCK = 1.

Bit 1 **FPUEREN**: FPU error break enable for FPUERRF
Set and cleared by software.
0: FPUERRF flag does not generate a break.
1: FPUERRF flag generates a break when asserted.
Note: This bit is write-protected when REGLCK = 1

Bit 0 **ACCEREN**: Access error break enable for ACCERRF
Set and cleared by software.
0: ACCERRF flag does not generate a break.
1: ACCERRF flag generates a break when asserted.
Note: This bit is write-protected when REGLCK = 1.

20.7.37 HSP break input configuration register (HSP_BKICFGR)

Address offset: 0x178

Reset value: 0x0000 0000

This register is used by the application to configure the break input. Write access operations to this register are not allowed when REGLCK is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSEN	Res.
														rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FSATEN	HDEGOVEN	FWEREN	Res.	Res.	Res.	Res.	PFCT31EN	PFCT30EN	PFCT29EN	PFCT28EN	Res.	OPCEREN	FPUEREN	ACCEREN
	rw	rw	rw					rw	rw	rw	rw		rw	rw	rw

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

Bit 17 **SSEN**: Single step enable
Set and cleared by software.
0: The single step function is disabled: the SPE is not frozen when the processing function transition detector provides a trigger.
1: The single step function is enabled: the SPE is frozen every time the processing function transition detector provides a trigger.
Note: This bit is write-protected when REGLCK = 1.

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **FSATEN**: FPU saturation break enable for FPUSATF
Set and cleared by software.
0: FPUSATF flag does not generate a break.
1: FPUSATF flag generates a break when asserted.
Note: This bit is write-protected when REGLCK = 1.

Bit 13 **HDEGOVEN**: HSP dedicated event generator overrun break enable for HDEGOVRF
Set and cleared by software.
0: HDEGOVRF flag does not generate a break.
1: HDEGOVRF flag generates a break when asserted.
Note: This bit is write-protected when REGLCK = 1

Bit 12 **FWEREN**: Firmware error break enable for FWERRF
Set and cleared by software.
0: FWERRF flag does not generate a break.
1: FWERRF flag generates a break when asserted.
Note: This bit is write-protected when REGLCK = 1.

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:4 **PFCTxEN**: Processing function flag break enable for PFCTF[x] (x = 31 to 28)
Set and cleared by software.
0: PFCTF[x] does not generate a break.
1: PFCTF[x] generates a break when asserted
Note: This bit is write-protected when REGLCK = 1.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **OPCEREN**: Opcode error break enable for OPCOERRF
Set and cleared by software.
0: OPCOERRF flag does not generate a break.
1: OPCOERRF flag generates a break when asserted.
Note: This bit is write-protected when REGLCK = 1.

Bit 1 **FPUEREN**: FPU error break enable for FPUERRF
Set and cleared by software.
0: FPUERRF flag does not generate a break.
1: FPUERRF flag generates a break when asserted.
Note: This bit is write-protected when REGLCK = 1.

Bit 0 **ACCEREN**: Access error break enable for ACCERRF
Set and cleared by software.
0: ACCERRF flag does not generate a break.
1: ACCERRF flag generates a break when asserted.
Note: This bit is write-protected when REGLCK = 1.

20.7.38 HSP SPE interrupt enable register (HSP_SPE_IER)

Address offset: 0x280

Reset value: 0x0000 0000

This register allows the SPE to enable or disable an interrupt source. This register is read-only for the CPU, only provided for debugging purposes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	H2CMFREEIE	C2HMRDYIE
														r	r

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

Bit 1 **H2CMFREEIE**: SPE to CPU message free interrupt enable
Set and cleared by software.
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **C2HMRDYIE**: CPU to SPE message ready interrupt enable
Set and cleared by software.
0: Interrupt disabled
1: Interrupt enabled

20.7.39 HSP SPE interrupt status register (HSP_SPE_ISR)

Address offset: 0x0284

Reset value: 0x0000 0000

This register provides the status flags handled by the HITC. This register is read-only for the CPU, only provided for debugging purposes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUF3EVTF	BUF2EVTF	BUF1EVTF	BUF0EVTF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTREQF	Res.	Res.	H2CMFREEF	C2HMRDYF
r	r	r	r								r			r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **BUFxEVTF**: BUFFx status (x = 3 to 0)
If configured in C2H direction, this bit is set by hardware when the SPE reads BUFFx. It is cleared when the CPU writes data in HSP_BUFFxDR.
If configured in H2C direction, this bit is set by hardware when the SPE writes BUFFx. It is cleared when the CPU reads a data from HSP_BUFFxDR register.
0: No event pending
1: An event is pending BUFFx empty (if set to C2H) or BUFFx not empty (if set to C2H).

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **RSTREQF**: HSP reset request
Set and cleared by software.
0: No pending reset request
1: A reset request from the application is pending.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **H2CMFREEF**: SPE to CPU message box status
Set and cleared by software.
0: The SPE to CPU message box is not free.
1: The SPE to CPU message box is free. A new message can be posted for the CPU.

Bit 0 **C2HMRDYF**: CPU to SPE message box status
Set and cleared by software.
0: No message from the CPU
1: A new message from the CPU is available.

20.7.40 HSP error interrupt enable register (HSP_ERR_IER)

Address offset: 0x398

Reset value: 0x0000 0000

This register controls the interrupt enables for all error events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DENORMIE	INVIE	DBZIE	OVFIE	UDFIE	ACCERRIE	OPCOERRIE	Res.	HDEGOVRIE	BKINIE	SCHERRIE	FWERRIE	CAPOVRIE
			rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B3ERRIE	B2ERRIE	B1ERRIE	B0ERRIE	Res.	Res.	Res.	TRGIOVRIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw				rw								

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **DENORMIE**: FPU denormal interrupt enable
Set and cleared by software. When this bit is enabled, FPUERRF is set when an invalid exception is reported by the FPU of the SPE.
0: Denormal exception does not activate the FPUERRF flag.
1: Denormal exception activates the FPUERRF flag.

Bit 27 **INVIE**: FPU Invalid exception interrupt enable
Set and cleared by software. When this bit is enabled, FPUERRF is set when an invalid exception is reported by the FPU of the SPE.
0: Invalid exception does not activate the FPUERRF flag.
1: Invalid exception activates the FPUERRF flag.

Bit 26 **DBZIE**: FPU division-by-zero exception interrupt enable
Set and cleared by software. When this bit is enabled, FPUERRF is set when a division-by-zero exception is reported by the FPU of the SPE.
0: Division-by-zero exception does not activate the FPUERRF flag.
1: Division-by-zero exception activates the FPUERRF flag.

Bit 25 **OVFIE**: FPU overflow exception interrupt enable
Set and cleared by software. When this bit is enabled, FPUERRF is set when an overflow exception is reported by the FPU of the SPE.
0: Overflow exception does not activate the FPUERRF flag.
1: Overflow exception activates the FPUERRF flag

Bit 24 **UDFIE**: FPU underflow exception interrupt enable
Set and cleared by software. When this bit is enabled, FPUERRF is set when an underflow exception is reported by the FPU of the SPE.
0: Underflow exception does not activate the FPUERRF flag.
1: Underflow exception activates the FPUERRF flag.

Bit 23 **ACCERRIE**: SPE access error interrupt enable
Set and cleared by software.
0: Access error interrupt disabled
1: Access error interrupt enabled

Bit 22 **OPCOERRIE**: Invalid OpCode error interrupt enable
Set and cleared by software.
0: OpCode error interrupt disabled
1: OpCode error interrupt enabled

Bit 21 Reserved, must be kept at reset value.

Bit 20 **HDEGOVRIE**: SPE event overrun interrupt enable
Set and cleared by software.
0: SPE event overrun interrupt disabled
1: SPE event overrun interrupt enabled

Bit 19 **BKINIE**: Break input interrupt enable
Set and cleared by software.
0: Break input interrupt disabled
1: Break input interrupt enabled

Bit 18 **SCHERRIE**: Scheduler error interrupt enable
Set and cleared by software.
0: Scheduler error interrupt disabled
1: Scheduler error interrupt enabled

Bit 17 **FWERRIE**: Firmware error interrupt enable
Set and cleared by software.
0: Firmware error interrupt disabled
1: Firmware error interrupt enabled

Bit 16 **CAPOVRIE**: Capture register overrun interrupt enable
Set and cleared by software.
0: Capture register overrun interrupt disabled
1: Capture register overrun interrupt enabled

Bits 15:12 **BxERRIE**: H2CBUFFx underrun or C2HBUFFx overrun interrupt enable (x = 3 to 0)
Set and cleared by software.
0: H2C/C2HBUFFx underrun/overrun interrupt disabled
1: H2C/C2HBUFFx underrun/overrun interrupt enabled

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGIOVRIE**: TRGITF overrun interrupt enable
Set and cleared by software.
0: TRGITF overrun interrupt disabled
1: TRGITF overrun interrupt enabled

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

20.7.41 HSP events interrupt enable register (HSP_EVT_IER)

Address offset: 0x39C

Reset value: 0x0000 0000

This register controls the interrupt enable for control events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FPUSATIE	Res.	Res.	TOVLPIE	Res.	Res.	Res.	Res.	EOTECP3IE	EOTECP2IE	EOTECP1IE	EOTECP0IE	SOTECP3IE	SOTECP2IE	SOTECP1IE	SOTECP0IE
rw			rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B3EVTIE	B2EVTIE	B1EVTIE	B0EVTIE	Res.	Res.	EOFWFEIE	SOFWFEIE	CDEGRDYIE	Res.	Res.	CAPRDYIE	Res.	DCDONEIE	C2HMFREEIE	H2CMRDYIE
rw	rw	rw	rw			rw	rw	rw			rw		rw	rw	rw

Bit 31 **FPUSATIE**: Saturation flag interrupt enable
Set by hardware, and cleared by software.
0: Saturation interrupt disabled
1: Saturation interrupt enabled

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **TOVLPIE**: Task overlap flag interrupt enable
Set by hardware, and cleared by software.
0: Task overlap interrupt disabled
1: Task overlap interrupt enabled

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **EOTECPxIE**: End of task generated by event comparator x interrupt enable (x = 3 to 0)
Set and cleared by software.
0: End of task interrupt generated by event comparator x disabled
1: End of task interrupt generated by event comparator x enabled

Bits 19:16 **SOTECPxIE**: Start of task generated by event comparator x interrupt enable (x = 3 to 0)
Set and cleared by software.
0: Start of task interrupt generated by event comparator x disabled
1: Start of task interrupt generated by event comparator x enabled

Bits 15:12 **BxEVTIE**: BUFFx event interrupt enable (x = 3 to 0)
Set and cleared by software. The buffer event is buffer empty for C2HBUFF, and buffer not empty for H2CBUFF.
0: BUFFx event interrupt disabled
1: BUFFx event interrupt enabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **EOFWFEIE**: End of WFE interrupt enable
Set and cleared by software.
0: End of WFE interrupt disabled
1: End of WFE interrupt enabled

Bit 8 **SOFWFEIE**: Start of WFE interrupt enable
Set and cleared by software.
0: Start of WFE interrupt disabled
1: Start of WFE interrupt enabled

Bit 7 **CDEGRDYIE**: CPU dedicated event generator ready interrupt enable
Set and cleared by software.
0: Dedicated software event generator ready interrupt disabled
1: Dedicated software event generator ready interrupt enabled

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **CAPRDYIE**: Capture buffer data ready interrupt enable
Set and cleared by software.
0: Capture buffer data ready interrupt disabled
1: Capture buffer data ready interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCDONEIE**: Direct command done interrupt enable
Set and cleared by software. This flag indicates that the execution of a direct command is completed.
0: End of direct command execution interrupt disabled
1: End of direct command execution interrupt enabled

Bit 1 **C2HMFREEIE**: CPU to HSP message box free interrupt enable
Set and cleared by software.
0: CPU to HSP message free interrupt disabled
1: CPU to HSP message free interrupt enabled

Bit 0 **H2CMRDYIE**: HSP to CPU message ready interrupt enable
Set and cleared by software.
0: HSP to CPU message ready interrupt disabled
1: HSP to CPU message ready interrupt enabled

20.7.42 HSP processing event interrupt enable register (HSP_PFCTEVT_IER)

Address offset: 0x3A0

Reset value: 0x0000 0000

This register controls the interrupt enable for flags/events generated by the SPE processing functions.

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

Bits 31:0 **PFCTIE[31:0]**: Processing function flags interrupt enable
Set and cleared by software. These flags are generated by the SPE processing functions.
0: Interrupt not generated on flag activation
1: Interrupt generated on flag activation

20.7.43 HSP error interrupt status register (HSP_ERR_ISR)

Address offset: 0x3A4

Reset value: 0x0000 0000

This register shows the event status for all error events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPUERRF	ACCERRF	OPCOERRF	Res.	HDEGOVRF	BKINF	SCHERRF	FWERRF	CAPOVRF
							r	r	r		r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B3ERRF	B2ERRF	B1ERRF	B0ERRF	Res.	Res.	Res.	TRGIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r				r								

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **FPUERRF**: Computation error flag
Set by hardware, and cleared through the FPUERRC bit. This flag can be set if one of the enabled exception occurred: underflow, overflow, division-by-zero, invalid (see *HSP error information register (HSP_ERRINFR)* to get information on the root cause).
0: No computation error event detected
1: A computation error event detected

Bit 23 **ACCERRF**: SPE Access error flag
Set by hardware, and cleared through the ACCERRC bit. This bit is asserted when a SPE access error is detected (see *HSP error information register (HSP_ERRINFR)* to get information on the root cause).
0: No access error event detected
1: An access error event detected

Bit 22 **OPCOERRF**: Invalid OpCode error flag
Set by hardware, and cleared through the OPCOERRC bit.
0: No OpCode error event detected
1: An OpCode error event detected

Bit 21 Reserved, must be kept at reset value.

Bit 20 **HDEGOVRF**: SPE event overrun flag
Set by hardware, and cleared through HDEGOVRC bit.
0: No SPE event overrun event detected
1: An SPE event overrun event detected

Bit 19 **BKINF**: Break input flag
Set by hardware, and cleared through the BKINC bit.
0: No break input event detected
1: A break input event detected

Bit 18 **SCHERRF**: Scheduler error flag
Set by hardware, and cleared through the SCHERRC bit.
0: No scheduler error detected
1: A scheduler error detected

Bit 17 **FWERRF**: Firmware error flag
Set by hardware, and cleared through the FWERRC bit.
0: No firmware error detected
1: A firmware error detected (see *HSP firmware error register (HSP_FWERR)* to get the firmware error number)

Bit 16 **CAPOVRF**: Capture register overrun flag
Set by hardware, and cleared through the CAPOVRC bit.
0: No capture register overrun event detected
1: A capture register overrun event detected

Bits 15:12 **BxERRF**: H2CBUFFx underrun or C2HBUFFx overrun flag (x = 3 to 0)
Set by hardware, and cleared through the B3ERRC bit.
0: No H2C/C2HBUFFx underrun/overrun event detected
1: A H2C/C2HBUFFx underrun/overrun event detected

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGIOVRF**: TRGITF overrun flag
Set by hardware, and cleared through the TRGIOVRC bit (see *HSP error information register (HSP_ERRINFR)* to define which TRGIN block generated an overrun).
0: No TRGITF overrun event detected
1: An TRGITF overrun event detected

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

20.7.44 HSP events interrupt status register (HSP_EVT_ISR)

Address offset: 0x3A8

Reset value: 0x0000 0000

This register shows the event status for all events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FPUSATF	Res.	Res.	TOVLPF	Res.	Res.	Res.	Res.	EOTECP3F	EOTECP2F	EOTECP1F	EOTECP0F	SOTECP3F	SOTECP2F	SOTECP1F	SOTECP0F
r			r					r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B3EVTF	B2EVTF	B1EVTF	B0EVTF	Res.	Res.	EOFWFEF	SOFWFEF	CDEGRDYF	Res.	Res.	CAPRDYF	Res.	DCDONEF	C2HMFREEF	H2CMRDYF
r	r	r	r			r	r	r			r		r	r	r

Bit 31 **FPUSATF**: FPU saturation flag
Set by hardware, and cleared through the FPUSATC bit. This flag indicates that the FPU detected and performed a saturation.
0: No saturation detected
1: A saturation detected

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **TOVLPF**: Task overlap flag
Set by hardware, and cleared through the TOVLPC bit.
0: No task overlap situation detected
1: A task overlap situation in the specified task range detected

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **EOTECPxF**: End of task flag, for event comparator x (x = 3 to 0)
Set by hardware, and cleared through the EOTECPxC bit. An end of task is detected by the event comparator x on falling edge of **tsk_cmp**[x].
0: No end of task event detected
1: An end of task event detected

Bits 19:16 **SOTECPxF**: Start of task flag, for event comparator x (x = 3 to 0)
Set by hardware, and cleared through the SOTECP3C bit. A start of task is detected by the event comparator x on rising edge of **tsk_cmp**[x].
0: No start of task event detected
1: A start of task event detected

Bits 15:12 **BxEVTF**: BUFFx event flag (x = 3 to 0)
Set by hardware, and cleared by software by writing C2HBUFFx or reading H2CBUFFx.
0: BUFFx not empty (if configured in C2H) or BUFFx empty if configured in H2C
1: BUFFx empty (if configured in C2H) or BUFFx not empty if configured in H2C

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **EOFWFEF**: End of WFE flag
Set by hardware and cleared by software through the EOFWFEC bit.
0: No end of WFE event detected
1: End of WFE event is detected

Bit 8 **SOFWFEF**: Start of WFE flag
Set by hardware and cleared by software through the SOFWFEC bit.
0: No start of WFE event detected
1: Start of WFE event detected

Bit 7 **CDEGRDYF**: CPU dedicated event generator ready flag
Set by hardware and cleared by software. This flag informs the CPU that the CDEG is ready to accept the generation of a new event. This flag goes to 1 when CDEGBSY goes from 1 to 0.
0: CDEG not ready
1: CDEG ready to accept the generation of a new event

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **CAPRDYF**: Capture buffer data ready flag
Set by hardware, and cleared when HSP_CAPDR is read.
0: No capture buffer data ready event detected
1: A capture buffer data ready event detected

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCDONEF**: Direct command done flag
Set by SPE, and cleared through the DCDONEC bit. This flag indicates that the execution of a direct command is completed.
0: No direct command under execution, or current execution not yet completed
1: Direct command execution completed

Bit 1 **C2HMFREEF**: CPU to HSP message box free flag
Set by hardware, and cleared through the C2HMFREEC bit.
0: No CPU to HSP message busy event detected
1: A CPU to HSP message busy event detected

Bit 0 **H2CMRDYF**: HSP to CPU message ready flag
Set by hardware, and cleared through the H2CMRDYC bit.
0: No HSP to CPU message ready event detected
1: A HSP to CPU message ready event detected

20.7.45 HSP processing event interrupt status register (HSP_PFCTEVT_ISR)

Address offset: 0x3AC

Reset value: 0x0000 0000

This register shows the flag status returned by the SPE processing functions.

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

Bits 31:0 **PFCTF[31:0]**: Flags generated by SPE processing functions
Set by SPE processing, cleared through the PFCTC[31:0] bits. For eah PFCTFn:
0: Flag not active
1: Flag active

20.7.46 HSP error interrupt clear register (HSP_ERR_ICR)

Address offset: 0x3B4

Reset value: 0x0000 0000

This register is used by the CPU to clear the status flags for error events. Writing 0 has no effect, writing 1 clears the corresponding flag in the HSP_ERR_ISR register. Reading returns zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPUERRC	ACCERRC	OPCOERRC	Res.	HDEGOVRC	BKINC	SCHERRC	FWERRC	CAPOVRC
							w	w	w		w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B3ERRC	B2ERRC	B1ERRC	B0ERRC	Res.	Res.	Res.	TRGIOVRC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w	w				w								

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **FPUERRC**: FPUERRF flag clear
0: No effect
1: Clears FPUERRF in HSP_ERR_ISR.

Bit 23 **ACCERRC**: ACCERRF flag clear
0: No effect
1: Clears ACCERRF in HSP_ERR_ISR, and also SAHBERR, MAHBERR, STIERR, DMERR, and CMERR in HSP_ERRINFR.

Bit 22 **OPCOERRC**: OPCOERRF flag clear
0: No effect
1: Clears OPCOERRF in HSP_ERR_ISR.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **HDEGOVRC**: HDEGOVRF flag clear
0: No effect
1: Clears HDEGOVRF in HSP_ERR_ISR.

Bit 19 **BKINC**: BKINF flag clear
0: No effect
1: Clears BKINF in HSP_ERR_ISR.

Bit 18 **SCHERRC**: SCHERRF flag clear
0: No effect
1: Clears SCHERRF in HSP_ERR_ISR.

Bit 17 **FWERRC**: FWERRF flag clear
0: No effect
1: Clears FWERRF in HSP_ERR_ISR.

Bit 16 **CAPOVRC**: CAPOVRF flag clear
0: No effect
1: Clears CAPOVRF in HSP_ERR_ISR.

Bit 15 **B3ERRC**: B3ERRF flag clear
0: No effect
1: Clears B3ERRF in HSP_ERR_ISR.

Bit 14 **B2ERRC**: B2ERRF flag clear
0: No effect
1: Clears B2ERRFin HSP_ERR_ISR.

Bit 13 **B1ERRC**: B1ERRF flag clear
0: No effect
1: Clears B1ERRF in HSP_ERR_ISR.

Bit 12 **B0ERRC**: B0ERRF flag clear
0: No effect
1: Clears B0ERRF in HSP_ERR_ISR.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGIOVRC**: TRGIOVRF flag clear
0: No effect
1: Clears TRGIOVRF in HSP_ERR_ISR.

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

20.7.47 HSP events interrupt clear register (HSP_EVT_ICR)

Address offset: 0x3B8

Reset value: 0x0000 0000

This register shows the event status for all events. Writing 0 has no effect. Writing 1 clears the corresponding flag in HSP_EVT_ISR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FPUSATC	Res.	Res.	TOVLPC	Res.	Res.	Res.	Res.	EOTECP3C	EOTECP2C	EOTECP1C	EOTECP0C	SOTECP3C	SOTECP2C	SOTECP1C	SOTECP0C
w			w					w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EOFWFEC	SOFWFEC	CDEGRDYC	Res.	Res.	Res.	Res.	DCDONEC	C2HMFREEC	H2CMRDYC
						w	w	w					w	w	w

Bit 31 **FPUSATC**: FPUSATF flag clear
0: No effect
1: Clears FPUSATF in HSP_EVT_ISR.

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **TOVLPC**: TOVLPF flag clear
0: No effect
1: Clears TOVLPF in HSP_EVT_ISR.

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **EOTECPxC**: EOTECPxF flag clear (x = 3 to 0)
0: No effect
1: Clears EOTECPxF in HSP_EVT_ISR.

Bits 19:16 **SOTECPxC**: SOTECPxF flag clear (x = 3 to 0)
0: No effect
1: Clears SOTECP3F in HSP_EVT_ISR.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **EOFWFEC**: EOFWFEF flag clear
0: No effect
1: Clears EOFWFEF in HSP_EVT_ISR.

Bit 8 **SOFWFEC**: SOFWFEF flag clear
0: No effect
1: Clears SOFWFEFF in HSP_EVT_ISR.

Bit 7 **CDEGRDYC**: CDEGRDYF flag clear
0: No effect
1: Clears CDEGRDYF in HSP_EVT_ISR.

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **DCDONEC**: DCDONEF flag clear
0: No effect
1: Clears DCDONEF in HSP_EVT_ISR.

Bit 1 **C2HMFREEC**: C2HMFREEF flag clear
0: No effect
1: Clears C2HMFREEF in HSP_EVT_ISR.

Bit 0 **H2CMRDYC**: H2CMRDYF flag clear
0: No effect
1: Clears H2CMRDYF in HSP_EVT_ISR.

20.7.48 HSP processing event interrupt clear register (HSP_PFCTEVT_ICR)

Address offset: 0x3BC

Reset value: 0x0000 0000

This register is used to clear the flag status from SPE processing functions. Writing 0 has no effect. Writing 1 clears the corresponding flag in HSP_PFCTEVT_ISR.

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

Bits 31:0 **PFCTC[31:0]**: PFCTF[31:0] flag clear
For each PFCTCn bit:
0: No effect
1: Clears PFCTFn in HSP_PFCTEVT_ISR.

20.7.49 HSP error information register (HSP_ERRINFR)

Address offset: 0x3B0

Reset value: 0x0000 0000

This register gives additional information on the error root cause in case of computation error, or SPE access error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TRGIERR[9:0]									
						r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	SAHBERR	MAHBERR	Res.	DMERR	CMERR	Res.	Res.	DENORM	INVALID	DIVZERO	OVFLOW	UDFLOW	INEXACT
			r	r		r	r			r	r	r	r	r	r

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

Bits 25:16 **TRGIERR[9:0]**: TRGIN error information
Updated by hardware when an overrun error occurs in one of the TRGINx. This bitfield is cleared by setting TRGIOVRC.
TRGIERR[x] = 0 means that TRGINx did not detect an overrun.
TRGIERR[x] = 1 means that TRGINx detected an overrun.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **SAHBERR**: Slave AHB access error flag

Updated by hardware when ACCERRF is set. This bit indicates the root cause of the ACCERRF assertion. This bit is cleared by setting ACCERRC.

0: The MMC did not detect invalid accesses from the device.

1: The MMC detected invalid accesses from the device. An invalid access is detected when the device performs an access to an invalid memory space, or when it accesses a memory area protected by CDRLCK[1:0]. Invalid read accesses return 0. Invalid write access operations are ignored.

Note: Accessing above BRAMAB_END (see Figure 160) also generates a bus error.

Bit 11 **MAHBERR**: Internal AHB access error flag

Updated by hardware when ACCERRF is set. This bit indicates the root cause of the ACCERRF assertion. This bit is cleared by setting ACCERRC.

0: The SPE did not generate an AHB access error.

1: The SPE generates an AHB error when accessing to HSP_CTRL.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **DMERR**: Data memory access error flag

Updated by hardware when ACCERRF is set. This bit indicates the root cause of the ACCERRF assertion. This bit is cleared by setting ACCERRC.

0: The SPE did not generate a DRAM/BRAM access error.

1: The SPE performed a load/store outside the DRAM/BRAM range.

Bit 8 **CMERR**: Code memory access error flag

Updated by hardware when ACCERRF is set. This bit indicates the root cause of the ACCERRF assertion. This bit is cleared by setting ACCERRC.

0: The SPE did not generate a CRAM/CROM access error..

1: The SPE performed a fetch/load outside the CRAM/CROM memory range

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **DENORM**: Denormal exception flag

Updated by hardware when a denormal number is detected. The application can clear this bit by writing FPUERRC to 1.

0: No Denormal exception detected

1: Denormal exception detected

Bit 4 **INVALID**: Invalid exception flag

Updated by hardware when an operation produces a result outside float32 domain (such as zero divided by zero or square root of a negative number). The application can clear this bit by writing FPUERRC to 1.

0: No Invalid exception detected

1: Invalid exception detected

Bit 3 **DIVZERO**: Division by zero exception flag

Updated by hardware when an operation produces a result beyond the finite range of float32 domain. The application can clear this bit by writing FPUERRC to 1.

0: No division by 0 exception detected

1: Division by 0 exception detected

Bit 2 **OVFLOW**: Overflow exception flag

Updated by hardware when an operation produces a result beyond the finite range of float32. The application can clear this bit by writing FPUERRC to 1.

0: No Overflow exception detected

1: Overflow exception detected

Bit 1 **UDFLOW**: Underflow exception flag

Updated by hardware when an operation produces a result closer to 0 than supported by the float32. The application can clear this bit by writing FPUERRC to 1.

0: No Underflow exception detected

1: Underflow exception detected

Bit 0 **INEXACT**: Inexact exception flag

Updated by hardware when an operation produces a result that does not fit into its intended destination. The result had to be altered by rounding it off to fit. The application can clear this bit by writing FPUERRC to 1.

0: No Inexact exception detected

1: Inexact exception detected

20.7.50 HSP firmware error register (HSP_FWERR)

Address offset: 0x188

Reset value: 0x0000 0000

This register allows the CPU to check the firmware error number reported by the SPE. This register is read-only for the CPU.

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

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

Bits 9:0 **FWERRN[9:0]**: Firmware error number

Set and cleared by SPE. The SPE writes in this register the error root cause to generate an interrupt to the CPU if enabled (see *Table 145: Firmware error number*).

20.7.51 HSP parameter register x (HSP_PARAMRx)

Address offset: 0x194 + 0x4 * x, (x = 0 to 15)

Reset value: 0xXXXX XXXX

These registers are used by the application to provide command parameters to the HSP. They are not reset. The application must initialize them when sending commands to the HSP.

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

Bits 31:0 **PARAM[31:0]**: Parameter value

Set and cleared by the application, which uses these registers to provide the parameters of commands entered through MSGB or DCMD. These registers can be read and written by the application.

20.7.52 HSP firmware versions register (HSP_FWVERR)

Address offset: 0x3DC

Reset value: 0x0000 0000

This register gives the CROM and CRAM firmware versions. The values inside this register are valid only if BSTAT[3:0] differs from 0.

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **CRAMAJREV[3:0]**: Code RAM major revision
This bitfield returns the CRAM major revision

Bits 19:12 **CRAMINREV[7:0]**: Code RAM minor revision
This bitfield returns the CRAM minor revision

Bits 11:8 **CROMAJREV[3:0]**: Code ROM major revision
This bitfield returns the CROM major revision

Bits 7:0 **CROMINREV[7:0]**: Code ROM minor revision
This bitfield returns the CROM minor revision

20.7.53 HSP SPE version register (HSP_SPEVERR)

Address offset: 0x3E0

Reset value: 0x0000 0000

This register gives the SPE version and the product ID number. The values inside this register are valid only if BSTAT[3:0] differs from 0.

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

Bits 31:24 **PRODID[7:0]**: Product ID number
This bitfield returns the ID of the CRAM/CROM firmware.

Bits 23:0 **SPEVER[23:0]**: SPE version
This bitfield returns the SPE version

20.7.54 HSP register map

Table 154. HSP register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	HSP_CR	BSTAT[3:0]				Res.	Res.	BARB[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	RSTOK	RSTREQ	REGLCK	CDRLCK[1:0]		DCMDEVEN	Res.	Res.	Res.	Res.	SPECGDIS	MMCCGDIS	Res.	CTRLCGDIS	Res.	Res.	Res.	BOOTEN
	Reset value	0	0	0	0			0	0							0	0	0	0	0	0					0	0		0				0
0x004 - 0x03C	Reserved	Res.																															
0x040	HSP_C2HSEMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C2HSEM
	Reset value																																0
0x044	HSP_C2HMSGDR	C2HDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	HSP_H2CSEMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	H2CSEM
	Reset value																																0
0x04C	HSP_H2CMSGDR	H2CDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050 - 0x054	Reserved	Res.																															
0x058	HSP_DCMDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCBS
	Reset value																																0
0x05C	HSP_DCMDPTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDOG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTRF2	PTRF1	PTRF0
	Reset value																	0													0	0	0
0x060	HSP_DCMDIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDID[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064 to 0x06C	HSP_DCMDPTR0 to HSP_DCMDPTR2	PTCPU[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070 - 0x07C	Reserved	Res.																															
0x080	HSP_EVTENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVTEN[21:0]																						Res.
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 154. HSP register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x084	HSP_ESYNCENR	Res.	Res.	Res.	Res.	Res.	CDEGSYNCEN	Res.	Res.	Res.	EVTSYNCEN[21:0]																						Res.
	Reset value						0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x088	HSP_ESYNC0SRCR	Res.	Res.	EV8SSRC[1:0]		Res.	Res.	EV7SSRC[1:0]		Res.	Res.	EV6SSRC[1:0]		Res.	Res.	EV5SSRC[1:0]		Res.	Res.	EV4SSRC[1:0]		Res.	Res.	EV3SSRC[1:0]		Res.	Res.	EV2SSRC[1:0]		Res.	Res.	EV1SSRC[1:0]	
	Reset value			0	0			0	0			0	0			0	0			0	0			0	0			0	0			0	0
0x08C	HSP_ESYNC1SRCR	Res.	Res.	EV16SSRC[1:0]		Res.	Res.	EV15SSRC[1:0]		Res.	Res.	EV14SSRC[1:0]		Res.	Res.	EV13SSRC[1:0]		Res.	Res.	EV12SSRC[1:0]		Res.	Res.	EV11SSRC[1:0]		Res.	Res.	EV10SSRC[1:0]		Res.	Res.	EV9SSRC[1:0]	
	Reset value			0	0			0	0			0	0			0	0			0	0			0	0			0	0			0	0
0x090	HSP_ESYNC2SRCR	CDEGSSRC[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EV22SSRC[1:0]		Res.	Res.	EV21SSRC[1:0]		Res.	Res.	EV20SSRC[1:0]		Res.	Res.	EV19SSRC[1:0]		Res.	Res.	EV18SSRC[1:0]		Res.	Res.	EV17SSRC[1:0]	
	Reset value	0	0									0	0			0	0			0	0			0	0			0	0			0	0
0x094	Reserved	Res.																															
0x098	HSP_PEVTLR	Res.	Res.	Res.	Res.	EVTL[27:0]																											
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x09C	Reserved	Res.																															
0x0A0	HSP_ITFENR	HDEGEN	HSEGEN	CDEGEN	CSEGEN	DCMDDIS	Res.	TRGI9EN	TRGI8EN	TRGI7EN	TRGI6EN	TRGI5EN	TRGI4EN	TRGI3EN	TRGI2EN	TRGI1EN	TRGI0EN	Res.	Res.	Res.	TRGOEN	Res.	Res.	Res.	STREAMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0				0				0								
0x0A4 - 0x0B0	Reserved	Res.																															
0x0B4	HSP_EVTSRC0R	Res.	EVT8SRC[2:0]			Res.	EVT7SRC[2:0]			Res.	EVT6SRC[2:0]			Res.	EVT5SRC[2:0]			Res.	EVT4SRC[2:0]			Res.	EVT3SRC[2:0]			Res.	EVT2SRC[2:0]			Res.	EVT1SRC[2:0]		
	Reset value		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0
0x0B8	HSP_EVTSRC1R	Res.	EVT16SRC[2:0]			Res.	EVT15SRC[2:0]			Res.	EVT14SRC[2:0]			Res.	EVT13SRC[2:0]			Res.	EVT12SRC[2:0]			Res.	EVT11SRC[2:0]			Res.	EVT10SRC[2:0]			Res.	EVT9SRC[2:0]		
	Reset value		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0

Table 154. HSP register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0BC	HSP_EVTSRC2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVT22SRC[2:0]			Res.	EVT21SRC[2:0]			Res.	EVT20SRC[2:0]			Res.	EVT19SRC[2:0]			Res.	EVT18SRC[2:0]			Res.	EVT17SRC[2:0]		
	Reset value										0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0
0x0C0 - 0x0F0	Reserved	Res.																															
0x0F4	HSP_BUFFCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMB2	COMB1	COMB0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2FEN	Res.	Res.	Res.	Res.	BUFF3DIR	BUFF2DIR	BUFF1DIR	BUFF0DIR
	Reset value														0	0	0								0					0	0	0	0
0x0F8 to 0x104	HSP_BUFF0DR to HSP_BUFF3DR	BUFFDAT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108 - 0x11C	Reserved	Res.																															
0x120	HSP_TRGINCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRG9POL	TRG8POL	TRG7POL	TRG6POL	TRG5POL	TRG4POL	TRG3POL	TRG2POL	TRG1POL	TRG0POL
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x124	HSP_TRGOCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGO3SRC[1:0]		TRGO2SRC[1:0]		TRGO1SRC[1:0]		TRGO0SRC[1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x128	Reserved	Res.																															
0x12C	HSP_CSEGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSEVT[21:0]																						Res.
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x130	HSP_CDEGR	CDEGBSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTSKN[5:0]					
	Reset value	0																										0	0	0	0	0	0
0x134 - 0x14C	Reserved	Res.																															
0x150	HSP_TRGINSELR0	Res.	Res.	TRG3SEL[5:0]						Res.	Res.	TRG2SEL[5:0]						Res.	Res.	TRG1SEL[5:0]						Res.	Res.	TRG0SEL[5:0]					
	Reset value			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x154	HSP_TRGINSELR1	Res.	Res.	TRG7SEL[5:0]						Res.	Res.	TRG6SEL[5:0]						Res.	Res.	TRG5SEL[5:0]						Res.	Res.	TRG4SEL[5:0]					
	Reset value			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x158	HSP_TRGINSELR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRG9SEL[5:0]						Res.	Res.	TRG8SEL[5:0]					
	Reset value																			0	0	0	0	0	0			0	0	0	0	0	0
0x15C - 0x164	Reserved	Res.																															

Table 154. HSP register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x168 to 0x174	HSP_BKO0CFGR to HSP_BKO3CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDEGOVEN	FWEREN	Res.	Res.	Res.	Res.	PFCT31EN	PFCT30EN	PFCT29EN	PFCT28EN	Res.	OPCEREN	FPUEREN	ACCEREN
	Reset value																			0	0					0	0	0	0		0	0	0
0x178	HSP_BKICFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSEN	Res.	Res.	FSATEN	HDEGOVEN	FWEREN	Res.	Res.	Res.	Res.	PFCT31EN	PFCT30EN	PFCT29EN	PFCT28EN	Res.	OPCEREN	FPUEREN	ACCEREN
	Reset value															0			0	0	0					0	0	0	0		0	0	0
0x17C - 0x184	Reserved	Res.																															
0x188	HSP_FWERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FWERRN[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x18C - 0x190	Reserved	Res.																															
0x194 to 0x1D0	HSP_PARAMR0 to HSP_PARAMR15	PARAM[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x1D4 - 0x27C	Reserved	Res.																															
0x280	HSP_SPE_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	H2CMFREEIE	C2HMRDYIE
	Reset value																														0	0	
0x284	HSP_SPE_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUF3EVTF	BUF2EVTF	BUF1EVTF	BUF0EVTF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTREQF	Res.	Res.	H2CMFREEF	C2HMRDYF
	Reset value																	0	0	0	0								0			0	0
0x288 - 0x33C	Reserved	Res.																															
0x340	HSP_TCUCFGR	TC3EN	Res.	TSKCMP3[5:0]						TC2EN	Res.	TSKCMP2[5:0]						TC1EN	Res.	TSKCMP1[5:0]						TC0EN	Res.	TSKCMP0[5:0]					
	Reset value	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0
0x344	HSP_TOVLPCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTNB[5:0]						Res.	Res.	LTNB[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOVLPEN
	Reset value											0	0	0	0	0	0			0	0	0	0	0	0								0
0x348 - 0x374	Reserved	Res.																															
0x378	HSP_SNPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNPSELB[2:0]			Res.	SNPSELA[2:0]		
	Reset value																										0	0	0		0	0	0
0x37C	HSP_CCNTR	CCNTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 154. HSP register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x380	HSP_CAPDR	TSKNB[5:0]						IFCNT[3:0]				IFHIST[7:0]								TSTAMP[13:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x384	HSP_CAPCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCNTREN	Res.	Res.	Res.	TSKFLT	Res.	Res.	Res.	PRESC[4:0]					Res.	FRCNTEN	CAPMOD[1:0]	
	Reset value																0				0				0	0	0	0	0		0	0	0
0x388 - 0x394	Reserved	Res.																															
0x398	HSP_ERR_IER	Res.	Res.	Res.	DENORMIE	INVIE	DBZIE	OVFIE	UDFIE	ACCERRIE	OPCOERRIE	Res.	HDEGOVRIE	BKINIE	SCHERRIE	FWERRIE	CAPOVRIE	B3ERRIE	B2ERRIE	B1ERRIE	B0ERRIE	Res.	Res.	Res.	TRGIOVRIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value				0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0				0								
0x39C	HSP_EVT_IER	FPUSATIE	Res.	Res.	TOVLPIE	Res.	Res.	Res.	Res.	EOTECP3IE	EOTECP2IE	EOTECP1IE	EOTECP0IE	SOTECP3IE	SOTECP2IE	SOTECP1IE	SOTECP0IE	B3EVTIE	B2EVTIE	B1EVTIE	B0EVTIE	Res.	Res.	EOFWFEIE	SOFWFEIE	CDEGRDYIE	Res.	Res.	CAPRDYIE	Res.	DCDONEIE	C2HMFREEIE	H2CMRDYIE
	Reset value	0			0					0	0	0	0	0	0	0	0	0	0	0	0			0	0	0			0		0	0	0
0x3A0	HSP_PFCTEVT_IER	PFCTIE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3A4	HSP_ERR_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPUERRF	ACCERRF	OPCOERRF	Res.	HDEGOVRF	BKINF	SCHERRF	FWERRF	CAPOVRF	B3ERRF	B2ERRF	B1ERRF	B0ERRF	Res.	Res.	Res.	TRGIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0	0	0		0	0	0	0	0	0	0	0	0				0								
0x3A8	HSP_EVT_ISR	FPUSATF	Res.	Res.	TOVLPF	Res.	Res.	Res.	Res.	EOTECP3F	EOTECP2F	EOTECP1F	EOTECP0F	SOTECP3F	SOTECP2F	SOTECP1F	SOTECP0F	B3EVTF	B2EVTF	B1EVTF	B0EVTF	Res.	Res.	EOFWFEF	SOFWFEF	CDEGRDYF	Res.	Res.	CAPRDYF	Res.	DCDONEF	C2HMFREEF	H2CMRDYF
	Reset value	0			0					0	0	0	0	0	0	0	0	0	0	0	0			0	0	0			0		0	0	0
0x3AC	HSP_PFCTEVT_ISR	PFCTF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3B0	HSP_ERRINFR	Res.	Res.	Res.	Res.	Res.	Res.	TRGIERR[9:0]										Res.	Res.	Res.	SAHBERR	MAHBERR	Res.	DMERR	CMERR	Res.	Res.	DENORM	INVALID	DIVZERO	OVFLOW	UDFLOW	INEXACT
	Reset value							0	0	0	0	0	0	0	0	0	0				0	0		0	0			0	0	0	0	0	0
0x3B4	HSP_ERR_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPUERRC	ACCERRC	OPCOERRC	Res.	HDEGOVRC	BKINC	SCHERRC	FWERRC	CAPOVRC	B3ERRC	B2ERRC	B1ERRC	B0ERRC	Res.	Res.	Res.	TRGIOVRC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0	0	0		0	0	0	0	0	0	0	0	0				0								
0x3B8	HSP_EVT_ICR	FPUSATC	Res.	Res.	TOVLPC	Res.	Res.	Res.	Res.	EOTECP3C	EOTECP2C	EOTECP1C	EOTECP0C	SOTECP3C	SOTECP2C	SOTECP1C	SOTECP0C	Res.	Res.	Res.	Res.	Res.	Res.	EOFWFEC	SOFWFEC	CDEGRDYC	Res.	Res.	Res.	Res.	DCDONEC	C2HMFREEC	H2CMRDYC
	Reset value	0			0					0	0	0	0	0	0	0	0							0	0	0					0	0	0
0x3BC	HSP_PFCTEVT_ICR	PFCTC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 154. HSP register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3DC	HSP_FWVERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRAMAJREV[3:0]				CRAMINREV[7:0]								CROMAJREV[3:0]				CROMINREV[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3E0	HSP_SPEVERR	PRODID[7:0]								SPEVER[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to (*Section 2.3.1: Introduction*) for the register boundary addresses.

21 Flexible memory controller (FMC)

The flexible memory controller (FMC) includes three memory controllers:

- The NOR/PSRAM memory controller
- The NAND memory controller
- The Synchronous DRAM (SDRAM/Mobile LPSDR SDRAM) controller

21.1 FMC main features

The FMC functional block makes the interface with: synchronous and asynchronous static memories, SDRAM memories, and NAND flash memory. Its main purposes are:

- to translate AXI transactions into the appropriate external device protocol
- to meet the access time requirements of the external memory devices

All external memories share the addresses, data and control signals with the controller. Each external device is accessed by means of a unique Chip Select. The FMC performs only one access at a time to an external device.

The main features of the FMC are the following:

- Interface with static-memory mapped devices including:
  - Static random access memory (SRAM)
  - NOR flash memory/OneNAND flash memory
  - PSRAM (4 memory banks)
  - NAND flash memory with ECC hardware to check up to 8 Kbytes of data
- Interface with synchronous DRAM (SDRAM/Mobile LPSDR SDRAM) memories
- Burst mode support for faster access to synchronous devices such as NOR flash memory, PSRAM and SDRAM)
- Programmable continuous clock output for asynchronous and synchronous accesses
- 8-,16-bit wide data bus
- Independent Chip Select control for each memory bank
- Independent configuration for each memory bank
- Write enable and byte lane select outputs for use with PSRAM, SRAM and SDRAM devices
- External asynchronous wait control
- Write FIFO with 16 x32-bit depth

  The Write FIFO is common to all memory controllers and consists of:
  - a Write Data FIFO which stores the data to be written to the memory
  - a Write Address FIFO which stores the address (up to 28 bits) plus the data size (up to 2 bits). When operating in Burst mode, only the start address is stored except when crossing a page boundary (for PSRAM and SDRAM). In this case, the burst is broken into two FIFO entries.
- Cacheable Read FIFO with 6 x64-bit depth (6 x14-bit address tag) for SDRAM controller.

At startup the FMC pins must be configured by the user application. The FMC I/O pins which are not used by the application can be used for other purposes.

The FMC registers that define the external device type and associated characteristics are set at boot time and do not change until the next reset or power-up. However, only a few bits can be changed on-the-fly:

- ECCEN and PBKEN bits in the FMC_PCR register
- IFS, IRS and ILS bits in the FMC_SR register
- MODE[2:0], CTB1/CTB2, NRFS and MRD bits in the FMC_SDCMR register
- REIE and CRE bits in the FMC_SDRTR register.

Follow the below sequence to modify parameters while the FMC is enabled:

1. First disable the FMC to prevent further accesses to any memory controller while the register is modified.
2. Update all required configurations.
3. Enable the FMC again.

When the SDRAM controller is used, if the SDCLK Clock ratio or refresh rate has to be modified after initialization phase, the following procedure must be followed.

1. Put the SDRAM device in Self-refresh mode.
2. Disable the FMC by resetting the FMCEN bit in the FMC_BCR1 register.
3. Update the required parameters.
4. Enable the FMC once all parameters have been updated.
5. Then, send the Clock Configuration Enable command to exit Self-fresh mode.

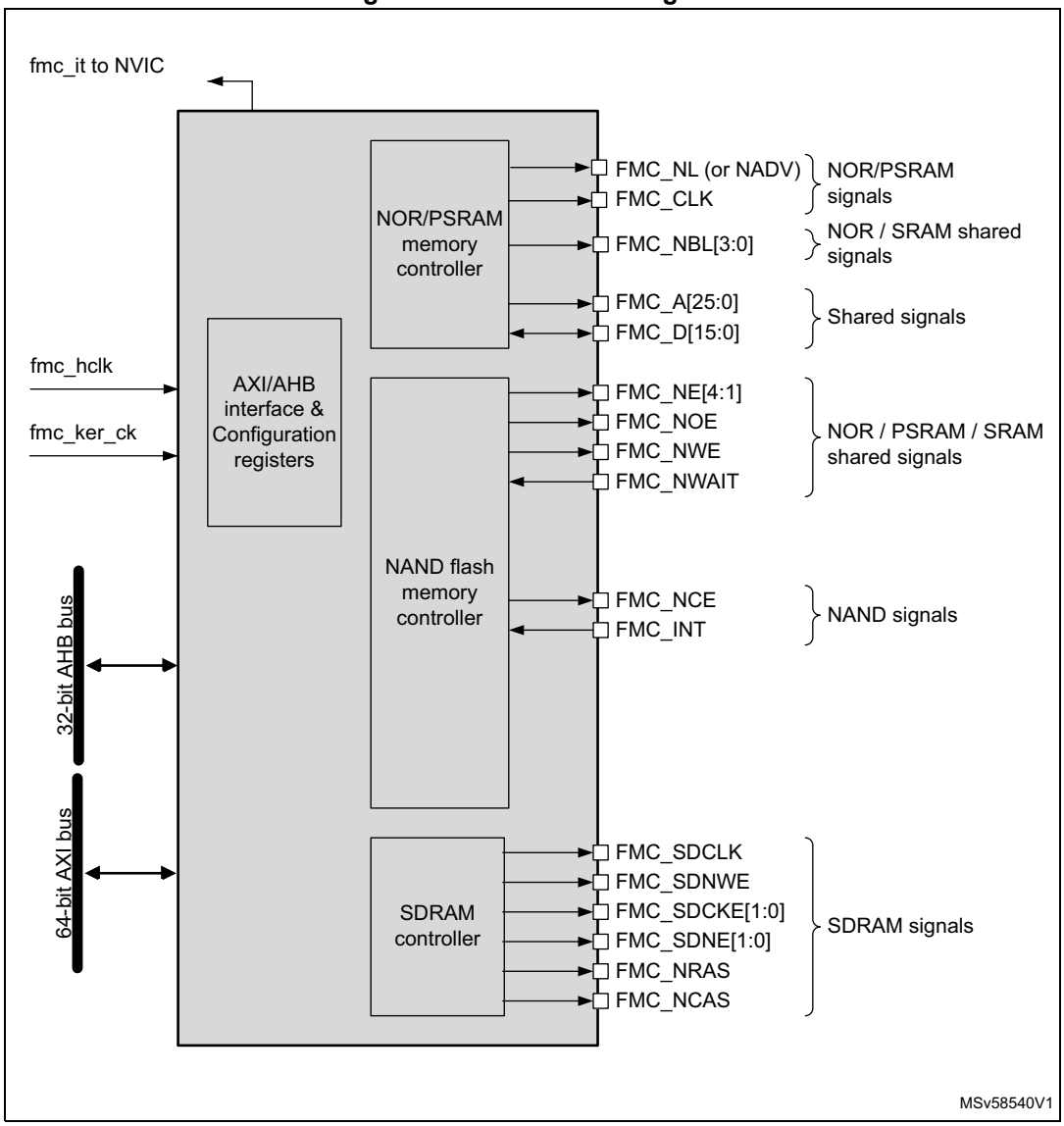
21.2 FMC block diagram

The FMC consists of the following main blocks:

- The NOR flash/PSRAM/SRAM controller
- The NAND controller
- The SDRAM controller
- The AXI interface
- The AHB interface (including the FMC configuration registers)

The block diagram is shown in the figure below.

Figure 200. FMC block diagram



21.3 FMC internal signals

Table 155 gives the list of FMC internal signals. FMC pins (or external signals) are described in *Section 21.7.1: External memory interface signals*.

Table 155. FMC pins

Names	Signal type	Description
fmc_it	Digital output	FMC interrupt
fmc_ker_ck	Digital input	FMC kernel clock
fmc_hclk	Digital input	FMC interface clock

21.4 AHB interface

The AHB slave interface allows internal CPUs to configure the FMC registers.

The AHB clock (fmc_hclk) is the reference clock for the FMC register accesses.

21.5 AXI interface

The AXI slave interface allows internal CPUs and other bus master peripherals to access the external memories.

AXI transactions are translated into the external device protocol. As the AXI data bus is 64-bit wide, the AXI transactions might be split into several consecutive 32-, 16- or 8-bit accesses according to data size accesses. The FMC Chip Select (FMC_NEx) does not toggle between consecutive accesses except in case of accesses in mode D when the Extended mode is enabled.

The FMC generates an AXI slave error when one of the following conditions is met:

- Reading or writing to an FMC bank (Bank 1 to 4) which is not enabled.
- Reading or writing to the NOR flash bank while the FACCEN bit is reset in the FMC_BCRx register.
- Writing to a write protected SDRAM bank (WP bit set in the FMC_SDCRx register).
- Violation of the SDRAM address range (access to reserved address range)
- Attempting to read/write access from/to SDRAM bank when it is not yet initialized

The FMC generates an AXI decoder error when ADDR[31:28] address bits are not supported by the FMC bank base address following the BMAP[1:0] bits configuration.

The kernel clock for the FMC is the asynchronous fmc_ker_ck clock (refer to for fmc_ker_ck clock source selection).

21.5.1 Supported memories and transactions

General transaction rules

The requested AXI transaction data size can be 8-, 16-, 32- or 64-bit wide whereas the accessed external device has a fixed data width. The best performance is always achieved with aligned AXI transactions whose size matches the external device data width.

When AXI transaction data size is different from the device data width, the result depends on the following factors:

- AXI transaction data size is greater than the device data width:
  - Read/Write transactions: the FMC splits the AXI transaction into smaller consecutive accesses matching the external device data width.
- AXI transaction data size is smaller than the external device data width and the device supports byte selection (SRAM, PSRAM, SDRAM):
  - Write transactions, the FMC manages the transaction using the byte lane signals.
  - Read transactions, the FMC returns all bytes according to the external device data width. The useless bytes are discarded by the system.
- AXI transaction data size is smaller than the external device data width and the device does not support byte selection (NOR and NAND flash memories):
  - Write transactions: the FMC writes some irrelevant bytes which may corrupt the external device
  - Read transactions: the FMC returns all bytes according to the external device data width. The useless bytes are discarded by the system.

Caution: Address alignment

- Read transactions with unaligned addresses (such as half-word starting at an odd address) are not supported by the FMC.
- Write transactions with unaligned addresses

  Their support depends on byte selection availability on the external device:
  - If the device does not support byte selection (NOR and NAND flash memories), narrow write transactions and/or unaligned write transaction are not supported since the FMC would write irrelevant bytes and corrupt the external device.

Wrap support for NOR flash/PSRAM and SDRAM

The synchronous memories must be configured in Linear burst mode of undefined length as not all masters can issue wrap transactions.

If a master generates a wrap transaction:

- The read is split into two linear burst transactions.
- The write is split into two linear burst transactions if the write FIFO is enabled and into several linear burst transactions if the write FIFO is disabled.

Configuration registers

The FMC can be configured through a set of registers. Refer to *Section 21.7.6*, for a detailed description of the NOR flash/PSRAM controller registers. Refer to *Section 21.8.7*, for a detailed description of the NAND flash registers and to *Section 21.9.5* for a detailed description of the SDRAM controller registers.

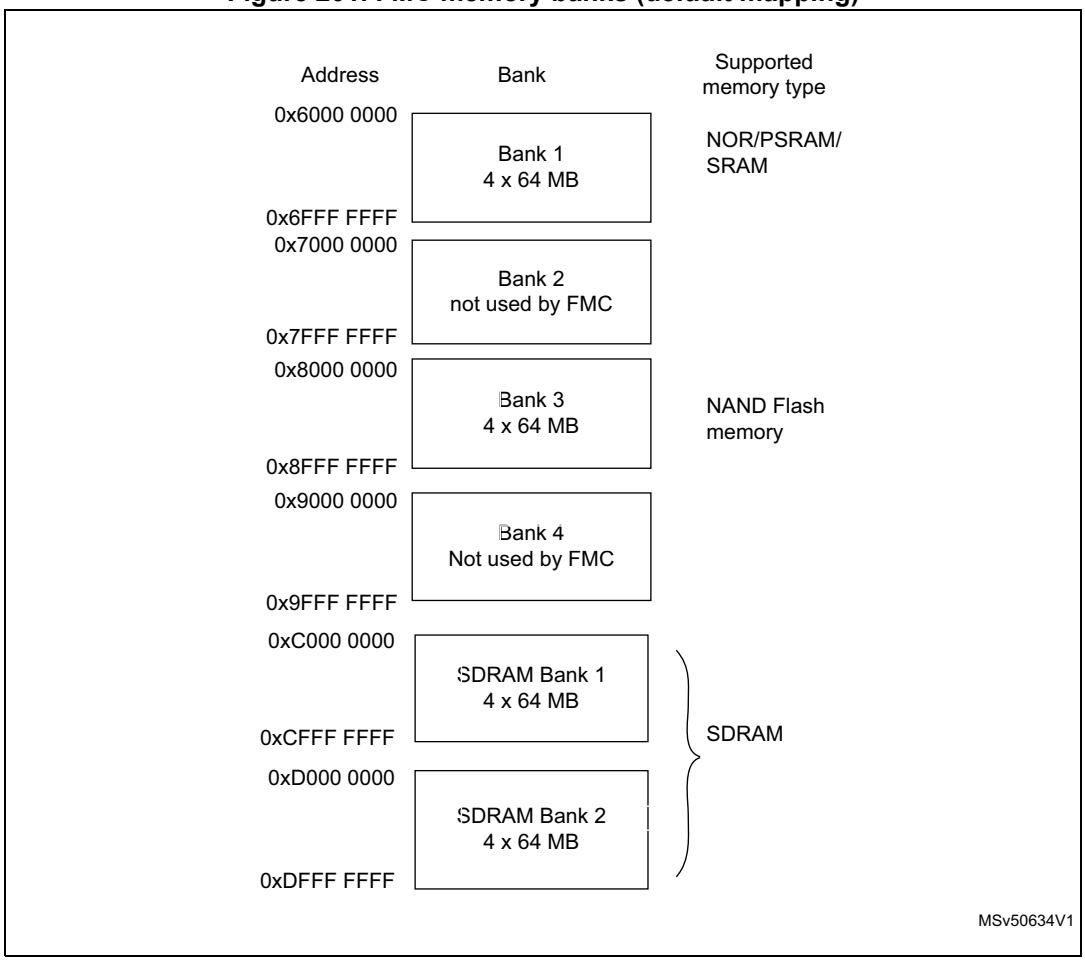
21.6 External device address mapping

From the FMC point of view, the external memory is divided into fixed-size banks of 256 Mbytes each (see *Figure 201*):

- Bank 1 is used to address up to 4 NOR flash memory or PSRAM devices. This bank is split into 4 NOR/PSRAM subbanks with 4 dedicated Chip Selects, as follows:
  - Bank 1 - NOR/PSRAM 1
  - Bank 1 - NOR/PSRAM 2
  - Bank 1 - NOR/PSRAM 3
  - Bank 1 - NOR/PSRAM 4
- Bank 2 and Bank 4 are not used by the FMC.
- Bank 3 is used to address NAND flash memory devices.The MPU memory attribute for this space must be reconfigured by software to Device.
- Bank 5 and 6 are used to address SDRAM devices (1 device per bank).

For each bank the type of memory to be used can be configured by the user application through the Configuration register.

Figure 201. FMC memory banks (default mapping)



The FMC bank mapping can be modified through the BMAP[1:0] bits in the FMC_BCR1 register. *Table 156* shows the configuration to swap the NOR/PSRAM bank with SDRAM banks.

Table 156. FMC bank mapping options

Start -End address	BMAP[1:0]=00 (Default mapping)	BMAP[1:0]=01 NOR/PSRAM and SDRAM banks swapped
0x6000 0000 - 0x6FFF FFFF	NOR/PSRAM bank	SDRAM bank1
0x7000 0000 - 0x7FFF FFFF	Not used by FMC	
0x8000 0000 - 0x8FFF FFFF	NAND bank	NAND bank
0x9000 0000 - 0x9FFF FFFF	Not used by FMC	
0xC000 0000 - 0xCFFF FFFF	SDRAM bank1	NOR/PSRAM bank
0xD000 0000 - 0xDFFF FFFF	SDRAM bank2	SDRAM bank2

21.6.1 NOR/PSRAM address mapping

ADDR[27:26] bits are used to select one of the four memory banks as shown in *Table 157*.

Table 157. NOR/PSRAM bank selection

ADDR[27:26](1)	Selected bank
00	Bank 1 - NOR/PSRAM 1
01	Bank 1 - NOR/PSRAM 2
10	Bank 1 - NOR/PSRAM 3
11	Bank 1 - NOR/PSRAM 4

1. ADDR are internal address lines that are translated to external memory.

The ADDR[25:0] bits contain the external memory address. Since ADDR is a byte address whereas the memory is addressed at word level, the address actually issued to the memory varies according to the memory data width, as shown in the following table.

Table 158. NOR/PSRAM External memory address

Memory width(1)	Data address issued to the memory	Maximum memory capacity (bits)
8-bit	ADDR[25:0]	64 Mbytes x 8 = 512 Mbit
16-bit	ADDR[25:1] >> 1	64 Mbytes/2 x 16 = 512 Mbit

1. In case of a 16-bit external memory width, the FMC internally uses ADDR[25:1] to generate the address for external memory FMC_A[24:0].
Whatever the external memory width, FMC_A[0] should be connected to external memory address A[0].

21.6.2 NAND flash memory address mapping

The NAND bank is divided into memory areas as indicated in *Table 159*.

Table 159. NAND memory mapping and timing registers

Start address	End address	FMC bank	Memory space	Timing register
0x8800 0000	0x8BFF FFFF	Bank 3 - NAND flash	Attribute	FMC_PATT (0x8C)
0x8000 0000	0x83FF FFFF		Common	FMC_PMEM (0x88)

For NAND flash memory, the common and attribute memory spaces are subdivided into three sections (see in *Table 160* below) located in the lower 256 Kbytes:

- Data section (first 64 Kbytes in the common/attribute memory space)
- Command section (second 64 Kbytes in the common / attribute memory space)
- Address section (next 128 Kbytes in the common / attribute memory space)

Table 160. NAND bank selection

Section name	ADDR[17:16]	Address range
Address section	1X	0x020000-0x03FFFF
Command section	01	0x010000-0x01FFFF
Data section	00	0x000000-0x0FFFF

The application software uses the 3 sections to access the NAND flash memory:

- **To send a command to NAND flash memory**, the software must write the command value to any memory location in the command section.
- **To specify the NAND flash address that must be read or written**, the software must write the address value to any memory location in the address section. Since an address can be 4 or 5 bytes long (depending on the actual memory size), several consecutive write operations to the address section are required to specify the full address.
- **To read or write data,** the software reads or writes the data from/to any memory location in the data section.

Since the NAND flash memory automatically increments addresses, there is no need to increment the address of the data section to access consecutive memory locations.

21.6.3 SDRAM address mapping

Two SDRAM banks are available as indicated in *Table 161*.

Table 161. SDRAM bank selection

Selected bank	Control register	Timing register
SDRAM Bank1	FMC_SDCR1	FMC_SDTR1
SDRAM Bank2	FMC_SDCR2	FMC_SDTR2

Table 162 shows SDRAM mapping for a 13-bit row and an 11-bit column configuration.

Table 162. SDRAM address mapping

Memory width(1)	Internal bank	Row address	Column address(2)	Maximum memory capacity (Mbytes)
8-bit	ADDR[25:24]	ADDR[23:11]	ADDR[10:0]	64 Mbytes: 4 x 8K x 2K
16-bit	ADDR[26:25]	ADDR[24:12]	ADDR[11:1]	128 Mbytes: 4 x 8K x 2K x 2

1. When interfacing with a 16-bit memory, the FMC internally uses the ADDR[11:1] internal address lines to generate the external address. When interfacing with a 32-bit memory, the FMC internally uses ADDR[12:2] lines to generate the external address. Whatever the memory width, FMC_A[0] has to be connected to the external memory address A[0].
2. The AutoPrecharge is not supported. FMC_A[10] must be connected to the external memory address A[10] but it is always driven low.

The ADDR[27:0] bits are translated into an external SDRAM address depending on the SDRAM controller configuration:

- Data size:8 or 16 bits
- Row size:11, 12 or 13 bits
- Column size: 8, 9, 10 or 11 bits
- Number of internal banks: two or four internal banks

The following tables show the SDRAM address mapping versus the SDRAM controller configuration.

Table 163. SDRAM address mapping with 8-bit data bus width(1)(2)

Row size configuration	ADDR(Internal Address Lines)																										
	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
11-bit row size configuration	Res.							Bank [1:0]	Row[10:0]												Column[7:0]							
	Res.								Bank [1:0]	Row[10:0]											Column[8:0]							
	Res.									Bank [1:0]	Row[10:0]										Column[9:0]							
	Res.										Bank [1:0]	Row[10:0]									Column[10:0]							
12-bit row size configuration	Res.								Bank [1:0]	Row[11:0]											Column[7:0]							
	Res.									Bank [1:0]	Row[11:0]										Column[8:0]							
	Res.										Bank [1:0]	Row[11:0]									Column[9:0]							
	Res.											Bank [1:0]	Row[11:0]								Column[10:0]							

Table 163. SDRAM address mapping with 8-bit data bus width(1)(2) (continued)

Row size configuration	ADDR(Internal Address Lines)																										
	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
13-bit row size configuration	Res.					Bank [1:0]			Row[12:0]												Column[7:0]							
	Res.					Bank [1:0]			Row[12:0]												Column[8:0]							
	Res.					Bank [1:0]			Row[12:0]												Column[9:0]							
	Res.					Bank [1:0]			Row[12:0]												Column[10:0]							

1. BANK[1:0] are the Bank Address BA[1:0]. When only 2 internal banks are used, BA1 must always be set to ‘0’.
2. Access to Reserved (Res.) address range generates an AXI slave error.

Table 164. SDRAM address mapping with 16-bit data bus width(1)(2)

Row size Configuration	ADDR(address Lines)																										
	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
11-bit row size configuration	Res.						Bank [1:0]		Row[10:0]												Column[7:0]							BM0 (3)
	Res.					Bank [1:0]			Row[10:0]												Column[8:0]							BM0
	Res.					Bank [1:0]			Row[10:0]												Column[9:0]							BM0
	Res.					Bank [1:0]			Row[10:0]												Column[10:0]							BM0
12-bit row size configuration	Res.					Bank [1:0]			Row[11:0]												Column[7:0]							BM0
	Res.					Bank [1:0]			Row[11:0]												Column[8:0]							BM0
	Res.					Bank [1:0]			Row[11:0]												Column[9:0]							BM0
	Res.					Bank [1:0]			Row[11:0]												Column[10:0]							BM0
13-bit row size configuration	Res.					Bank [1:0]			Row[12:0]												Column[7:0]							BM0
	Res.					Bank [1:0]			Row[12:0]												Column[8:0]							BM0
	Res.					Bank [1:0]			Row[12:0]												Column[9:0]							BM0
	Res.					Bank [1:0]			Row[12:0]												Column[10:0]							BM0

1. BANK[1:0] are the Bank Address BA[1:0]. When only 2 internal banks are used, BA1 must always be set to ‘0’.
2. Access to Reserved space (Res.) generates an AXI Slave error.
3. BM0: is the byte mask for 16-bit access.

Table 165. SDRAM address mapping with 32-bit data bus width(1)(2)

Row size configuration	ADDR(address Lines)																											
	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
11-bit row size configuration	Res.					Bank [1:0]		Row[10:0]											Column[7:0]								BM[1:0](3)	
	Res.					Bank [1:0]		Row[10:0]											Column[8:0]								BM[1:0	
	Res.					Bank [1:0]		Row[10:0]											Column[9:0]								BM[1:0	
	Res.					Bank [1:0]		Row[10:0]											Column[10:0]								BM[1:0	
12-bit row size configuration	Res.					Bank [1:0]		Row[11:0]											Column[7:0]								BM[1:0	
	Res.					Bank [1:0]		Row[11:0]											Column[8:0]								BM[1:0	
	Res.					Bank [1:0]		Row[11:0]											Column[9:0]								BM[1:0	
	Res.					Bank [1:0]		Row[11:0]											Column[10:0]								BM[1:0	
13-bit row size configuration	Res.					Bank [1:0]		Row[12:0]											Column[7:0]								BM[1:0	
	Res.					Bank [1:0]		Row[12:0]											Column[8:0]								BM[1:0	
	Res.					Bank [1:0]		Row[12:0]											Column[9:0]								BM[1:0	
	Bank [1:0]					Row[12:0]													Column[10:0]								BM[1:0	

1. BANK[1:0] are the Bank Address BA[1:0]. When only 2 internal banks are used, BA1 must always be set to ‘0’.
2. Access to Reserved space (Res.) generates an AXI slave error.
3. BM[1:0]: is the byte mask for 32-bit access.

21.7 NOR flash/PSRAM controller

The FMC generates the appropriate signal timings to drive the following types of memories:

- Asynchronous SRAM and ROM
  - 8 bits
  - 16 bits
- PSRAM (Cellular RAM)
  - Asynchronous mode
  - Burst mode for synchronous accesses with configurable option to split burst access when crossing boundary page for CRAM 1.5.
  - Multiplexed or non-multiplexed
- NOR flash memory
  - Asynchronous mode
  - Burst mode for synchronous accesses
  - Multiplexed or non-multiplexed

The FMC outputs a unique Chip Select signal, NE[4:1], per bank. All the other signals (addresses, data and control) are shared.

The FMC supports a wide range of devices through a programmable timings among which:

- Programmable wait states (up to 15)
- Programmable bus turnaround cycles (up to 15)
- Programmable output enable and write enable delays (up to 15)
- Independent read and write timings and protocol to support the widest variety of memories and timings
- Programmable continuous clock (FMC_CLK) output.

The FMC output Clock (FMC_CLK) is a sub-multiple of the fmc_ker_ck clock. It can be delivered to the selected external device either during synchronous accesses only or during asynchronous and synchronous accesses depending on the CCKEN bit configuration in the FMC_BCR1 register:

- If the CCLKEN bit is reset, the FMC generates the clock (FMC_CLK) only during synchronous accesses (Read/write transactions).
- If the CCLKEN bit is set, the FMC generates a continuous clock during asynchronous and synchronous accesses. To generate the FMC_CLK continuous clock, Bank 1 must be configured in Synchronous mode (see *Section 21.7.6: NOR/PSRAM controller registers*). Since the same clock is used for all synchronous memories, when a continuous output clock is generated and synchronous accesses are performed, the AXI data size has to be the same as the memory data width (MWID) otherwise the FMC_CLK frequency is changed depending on AXI data transaction (refer to *Section 21.7.5: Synchronous transactions* for FMC_CLK divider ratio formula).

The size of each bank is fixed and equal to 64 Mbytes. Each bank is configured through dedicated registers (see *Section 21.7.6: NOR/PSRAM controller registers*).

The programmable memory parameters include access times (see *Table 166*) and support for wait management (for PSRAM and NOR flash memory accessed in Burst mode).

Table 166. Programmable NOR/PSRAM access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Address setup	Duration of the address setup phase	Asynchronous	FMC clock cycle (fmc_ker_ck)	0	15
Address hold	Duration of the address hold phase	Asynchronous, muxed I/Os	FMC clock cycle (fmc_ker_ck)	1	15
Data setup	Duration of the data setup phase	Asynchronous	FMC clock cycle (fmc_ker_ck)	1	256
Bust turn	Duration of the bus turnaround phase	Asynchronous and synchronous read	FMC clock cycle (fmc_ker_ck)	0	15
Clock divide ratio	Number of FMC clock cycles (fmc_ker_ck) to build one memory clock cycle (CLK)	Synchronous	FMC clock cycle (fmc_ker_ck)	2	16
Data latency	Number of clock cycles to issue to the memory before the first data of the burst	Synchronous	Memory clock cycle (fmc_ker_ck)	2	17

21.7.1 External memory interface signals

Table 167, *Table 168* and *Table 169* list the signals that are typically used to interface with NOR flash memory, SRAM and PSRAM.

Note: The prefix “N” identifies the signals which are active low.

NOR flash memory, non-multiplexed I/Os

Table 167. Non-multiplexed I/O NOR flash memory

FMC pin name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:0]	O	Address bus
D[15:0]	I/O	Bidirectional data bus
NE[x]	O	Chip Select, x = 1..4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR flash devices)
NWAIT	I	NOR flash wait input signal to the FMC

The maximum capacity is 512 Mbits (26 address lines).

NOR flash memory, 16-bit multiplexed I/Os

Table 168. 16-bit multiplexed I/O NOR flash memory

FMC pin name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)
NE[x]	O	Chip Select, x = 1..4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR flash devices)
NWAIT	I	NOR flash wait input signal to the FMC

The maximum capacity is 512 Mbits.

PSRAM/SRAM, non-multiplexed I/Os

Table 169. Non-multiplexed I/Os PSRAM/SRAM

FMC pin name	I/O	Function
CLK	O	Clock (only for PSRAM synchronous access)
A[25:0]	O	Address bus
D[15:0]	I/O	Data bidirectional bus
NE[x]	O	Chip Select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid only for PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FMC
NBL[1:0]	O	Byte lane output. Byte 0 and Byte 1 control (upper and lower byte enable)

The maximum capacity is 512 Mbits.

PSRAM, 16-bit multiplexed I/Os

Table 170. 16-Bit multiplexed I/O PSRAM

FMC pin name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)
NE[x]	O	Chip Select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FMC
NBL[1:0]	O	Byte lane output. Byte 0 and Byte 1 control (upper and lower byte enable)

The maximum capacity is 512 Mbits (26 address lines).

21.7.2 Supported memories and transactions

Table 171 below shows an example of the supported devices, access modes and transactions when the memory data bus is 16-bit wide for NOR flash memory, PSRAM and SRAM. The transactions not allowed (or not supported) by the FMC are shown in gray in this example.

Table 171. NOR flash/PSRAM: Example of supported memories and transactions(1)

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
NOR flash (muxed I/Os and non-multiplexed I/Os)	Asynchronous	R	8	16	Y	
	Asynchronous	W	8	16	N	
	Asynchronous	R	16	16	Y	
	Asynchronous	W	16	16	Y	
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	
	Synchronous	R	16	16	Y	
	Synchronous	R	32/64	16	Y	
PSRAM (multiplexed I/Os and non-multiplexed I/Os)	Asynchronous	R	8	16	Y	
	Asynchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	16	16	Y	
	Asynchronous	W	16	16	Y	
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	
	Synchronous	R	16	16	Y	
	Synchronous	R	32/64	16	Y	
	Synchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Synchronous	W	16/32/64	16	Y	

Table 171. NOR flash/PSRAM: Example of supported memories and transactions(1) (continued)

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
SRAM and ROM	Asynchronous	R	8/16	16	Y	
	Asynchronous	W	8/16	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses Use of byte lanes NBL[1:0]
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses Use of byte lanes NBL[1:0]

1. NBL[1:0] are also driven by AXI write strobes.

21.7.3 General timing rules

Signal synchronization is performed as follows:

- All controller output signals change on the rising edge of the fmc_ker_ck clock.
- In Synchronous read and write modes, all output signals change on the rising edge of fmc_ker_ck clock. Whatever the CLKDIV value, all outputs change as follows:
  - NOEL/NWEL/ NEL/NADVL/ NADVH /NBLL/ Address valid outputs change on the falling edge of FMC_CLK clock.
  - NOEH/ NWEH / NEH/ NOEH/NBLH/ Address invalid outputs change on the rising edge of FMC_CLK clock.

21.7.4 NOR flash/PSRAM controller asynchronous transactions

Asynchronous transactions on static memories (NOR flash memory, PSRAM, SRAM) are performed as follows:

- Signals are synchronized by the internal clock. This clock is not issued to the memory.
- The FMC always samples the data before deasserting the Chip Select signal. This guarantees that the memory data hold timing constraint is met (minimum Chip Enable high to data transition is usually 0 ns)
- If the Extended mode is enabled (EXTMOD bit is set in the FMC_BCRx register), up to four extended modes (A, B, C and D) are available. It is possible to mix A, B, C and D modes for read and write operations. For example, read operation can be performed in mode A and write in mode B.
- If the Extended mode is disabled (EXTMOD bit is reset in the FMC_BCRx register), the FMC can operate in Mode1 or Mode2 as follows:
  - Mode 1 is the default mode when SRAM/PSRAM memory type is selected (MTYP = 0x0 or 0x01 in the FMC_BCRx register)
  - Mode 2 is the default mode when NOR memory type is selected (MTYP = 0x10 in the FMC_BCRx register).

Mode 1 - SRAM/PSRAM (CRAM)

The next figures show the read and write transactions for the supported modes followed by the required configuration of FMC_BCRx, and FMC_BTRx/FMC_BWTRx registers.

Figure 202. Mode 1 read access waveforms

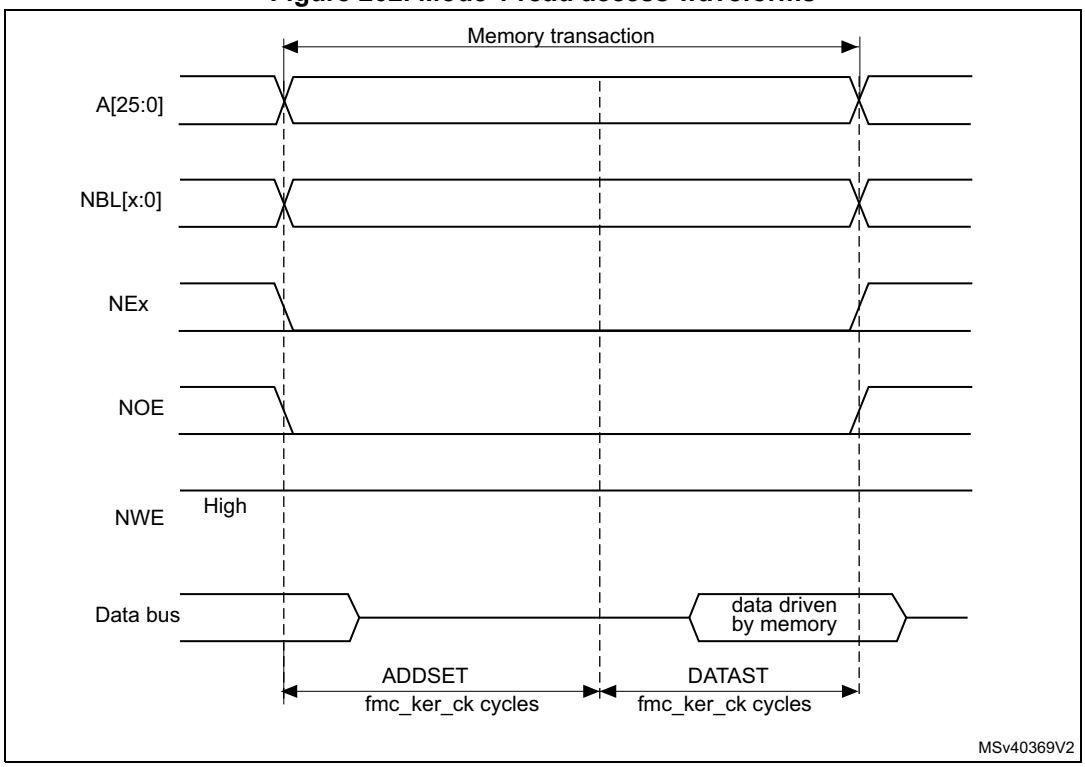
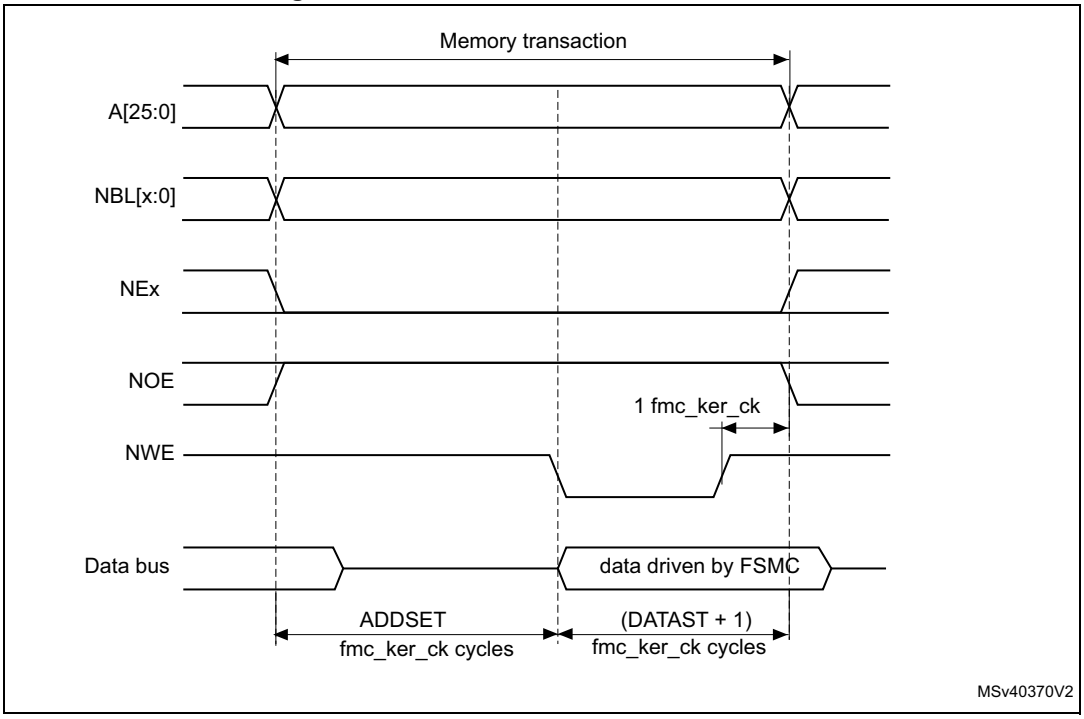


Figure 203. Mode 1 write access waveforms



The fmc_ker_ck cycle at the end of the write transaction helps guarantee the address and data hold time after the NWE rising edge. Due to the presence of this fmc_ker_ck cycle, the DATAST value must be greater than zero (DATAST > 0).

Table 172. FMC_BCRx bitfields (mode 1)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:26	Reserved	0x000
25:24	BMAP	As needed
23:22	Reserved	0x000
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0

Table 172. FMC_BCRx bitfields (mode 1) (continued)

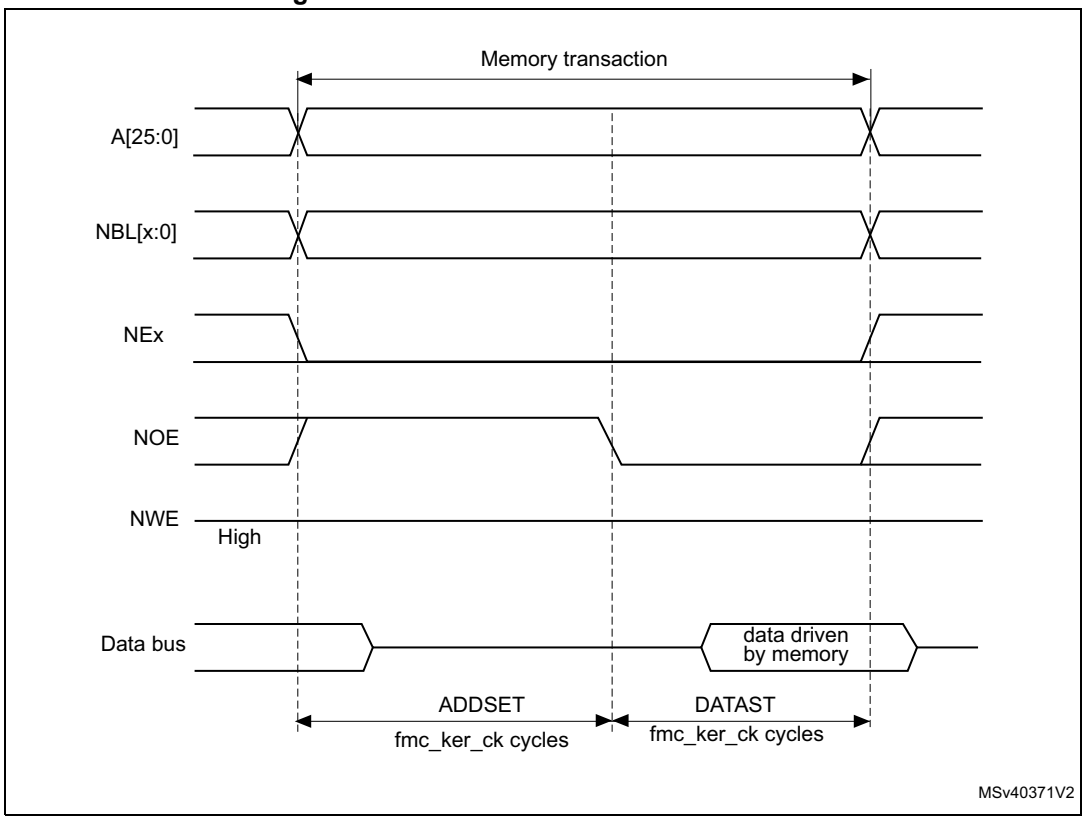
Bit number	Bit name	Value to set
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Don’t care
5:4	MWID	As needed
3:2	MTYP	As needed, exclude 0x2 (NOR flash memory)
1	MUXE	0x0
0	MBKEN	0x1

Table 173. FMC_BTRx bitfields (mode 1)

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	Don’t care
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the second access phase (DATAST+1 fmc_ker_ck cycles for write accesses, DATAST fmc_ker_ck cycles for read accesses).
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles). Minimum value for ADDSET is 0.

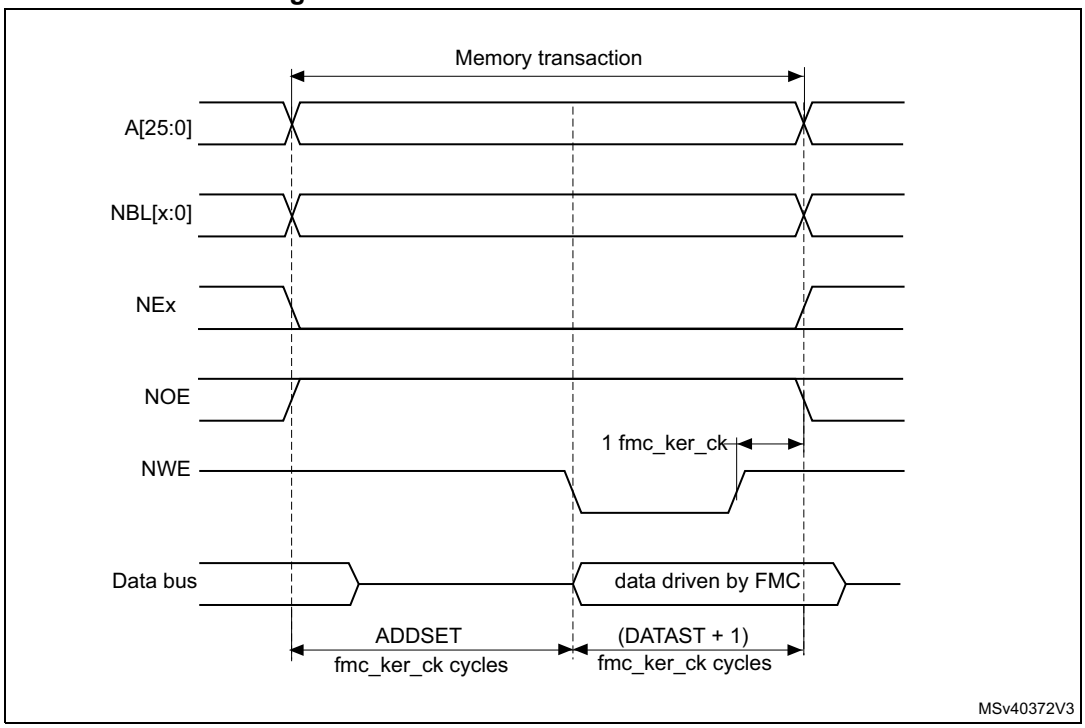
Mode A - SRAM/PSRAM (CRAM) OE toggling

Figure 204. Mode A read access waveforms



1. NBL[1:0] are driven low during the read access

Figure 205. Mode A write access waveforms



The differences compared with Mode1 are the toggling of NOE and the independent read and write timings.

Table 174. FMC_BCRx bitfields (mode A)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:26	Reserved	0x000
25:24	BMAP	As needed
23:22	Reserved	0x000
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0

Table 174. FMC_BCRx bitfields (mode A) (continued)

Bit number	Bit name	Value to set
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Don't care
5:4	MWID	As needed
3:2	MTYP	As needed, exclude 0x2 (NOR flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 175. FMC_BTRx bitfields (mode A)

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x0
27:24	DATLAT	Don't care
23:20	CLKDIV	Don't care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.
7:4	ADDHLD	Don't care
3:0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.

Table 176. FMC_BWTRx bitfields (mode A)

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x0
27:24	DATLAT	Don't care
23:20	CLKDIV	Don't care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for write accesses.
7:4	ADDHLD	Don't care
3:0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.

Mode 2/B - NOR flash

Figure 206. Mode 2 and mode B read access waveforms

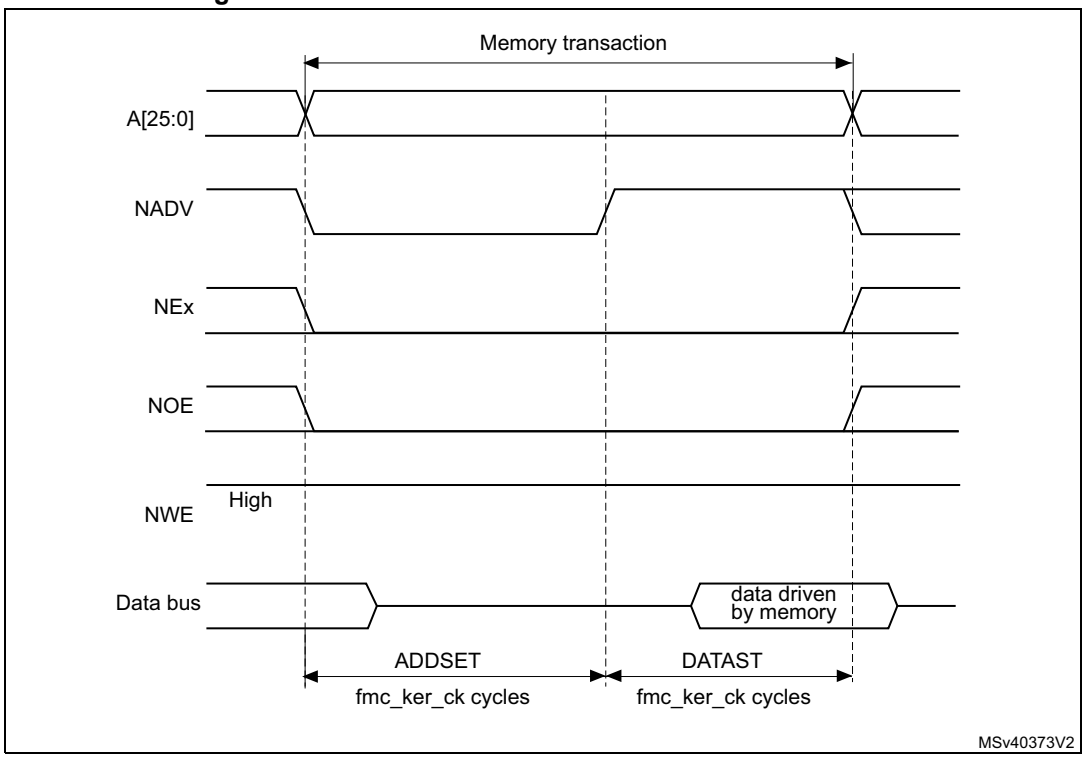


Figure 207. Mode 2 write access waveforms

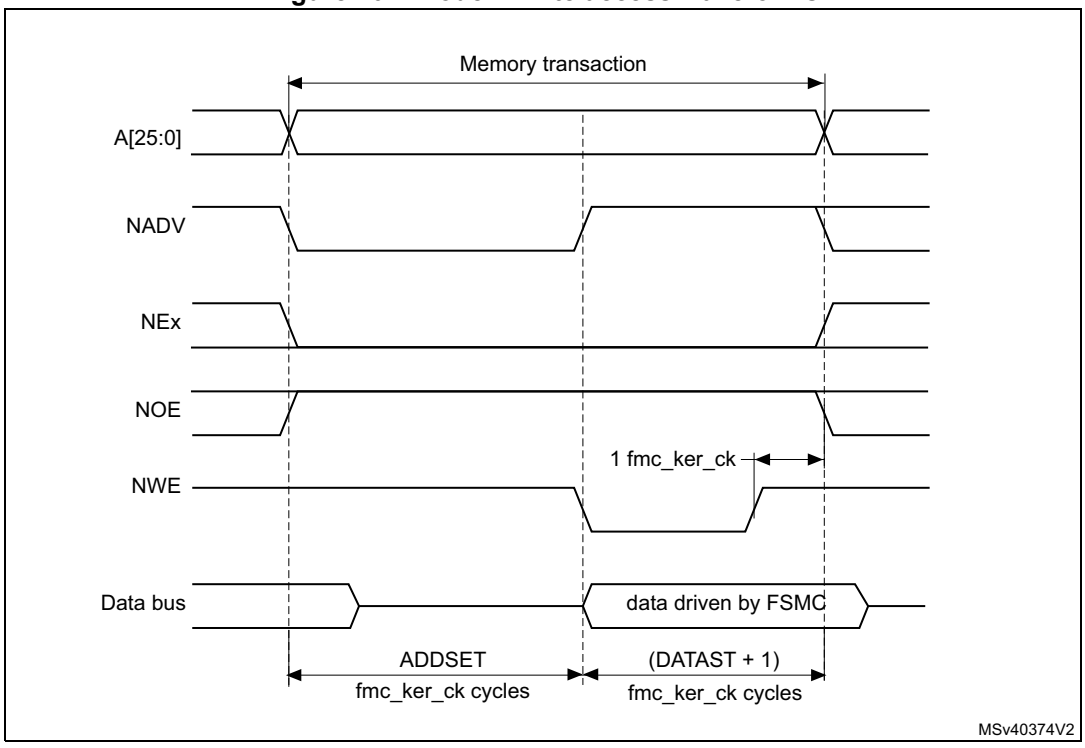
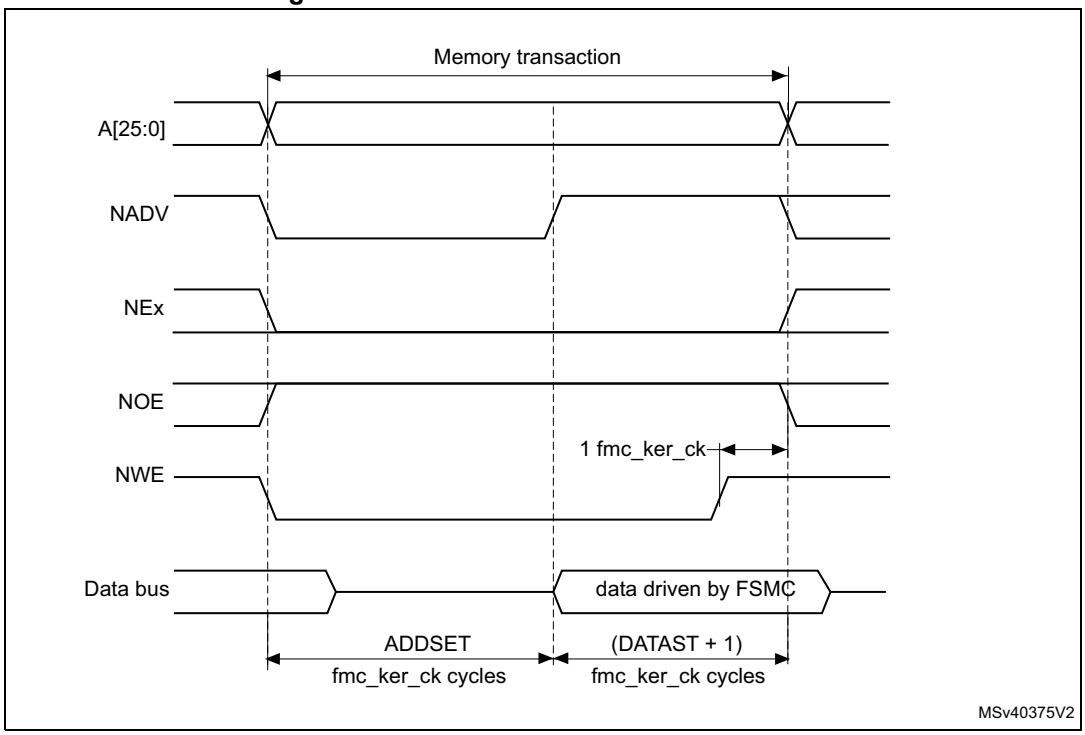


Figure 208. Mode B write access waveforms



The differences with Mode1 are the toggling of NWE and the independent read and write timings when extended mode is set (mode B).

Table 177. FMC_BCRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:26	Reserved	0x000
25:24	BMAP	As needed
23:22	Reserved	0x000
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1 for mode B, 0x0 for mode 2
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0

Table 177. FMC_BCRx bitfields (mode 2/B) (continued)

Bit number	Bit name	Value to set
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5:4	MWID	As needed
3:2	MTYP	0x2 (NOR flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 178. FMC_BTRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x1 if Extended mode is set
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the access second phase (DATAST fmc_ker_ck cycles) for read accesses.
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the access first phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.

Table 179. FMC_BWTRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x1 if Extended mode is set
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the access second phase (DATAST fmc_ker_ck cycles) for write accesses.
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the access first phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.

Note: The FMC_BWTRx register is valid only if the Extended mode is set (mode B), otherwise its content is don't care.

Mode C - NOR flash - OE toggling

Figure 209. Mode C read access waveforms

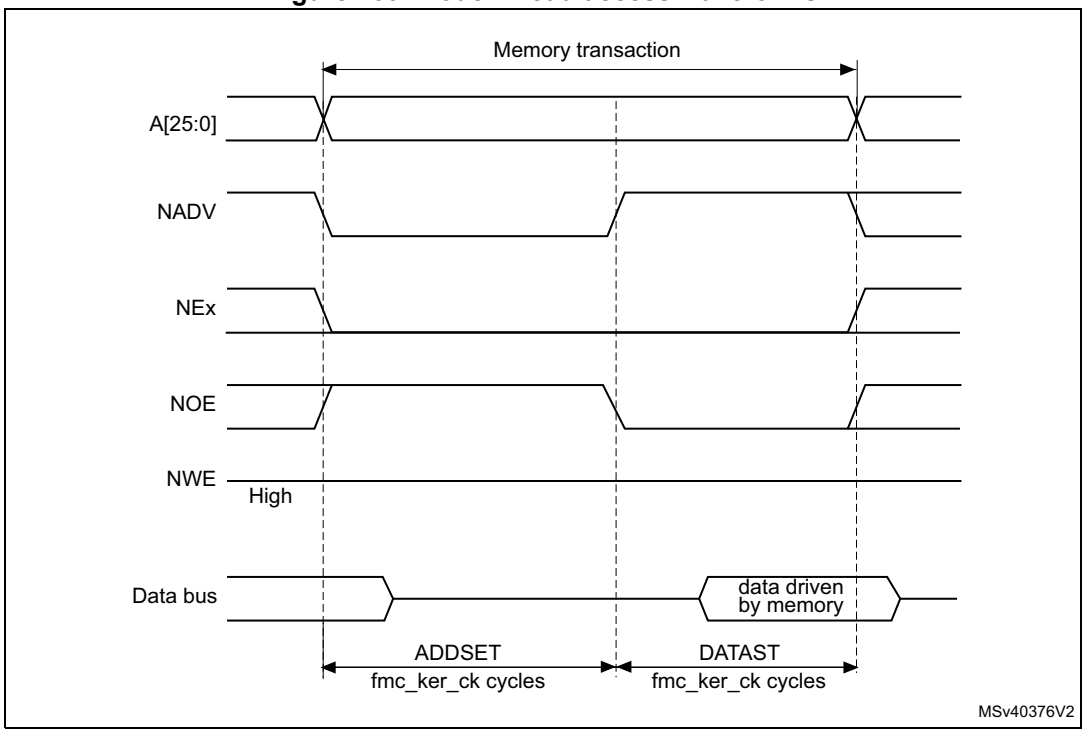
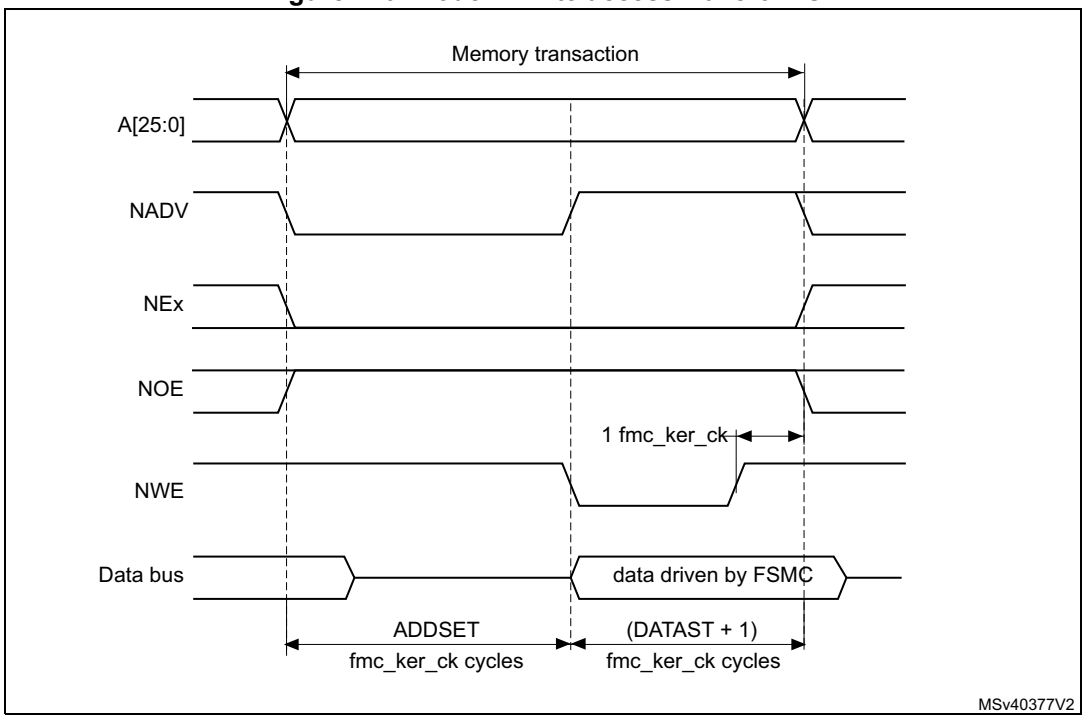


Figure 210. Mode C write access waveforms



The differences compared with Mode1 are the toggling of NOE and the independent read and write timings.

Table 180. FMC_BCRx bitfields (mode C)

Bit No.	Bit name	Value to set
31	FMCEN	0x1
30:26	Reserved	0x000
25:24	BMAP	As needed
23:22	Reserved	0x000
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5:4	MWID	As needed
3:2	MTYP	0x02 (NOR flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 181. FMC_BTRx bitfields (mode C)

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x2
27:24	DATLAT	0x0
23:20	CLKDIV	0x0
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.

Table 182. FMC_BWTRx bitfields (mode C)

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x2
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for write accesses.
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.

Mode D - asynchronous access with extended address

Figure 211. Mode D read access waveforms

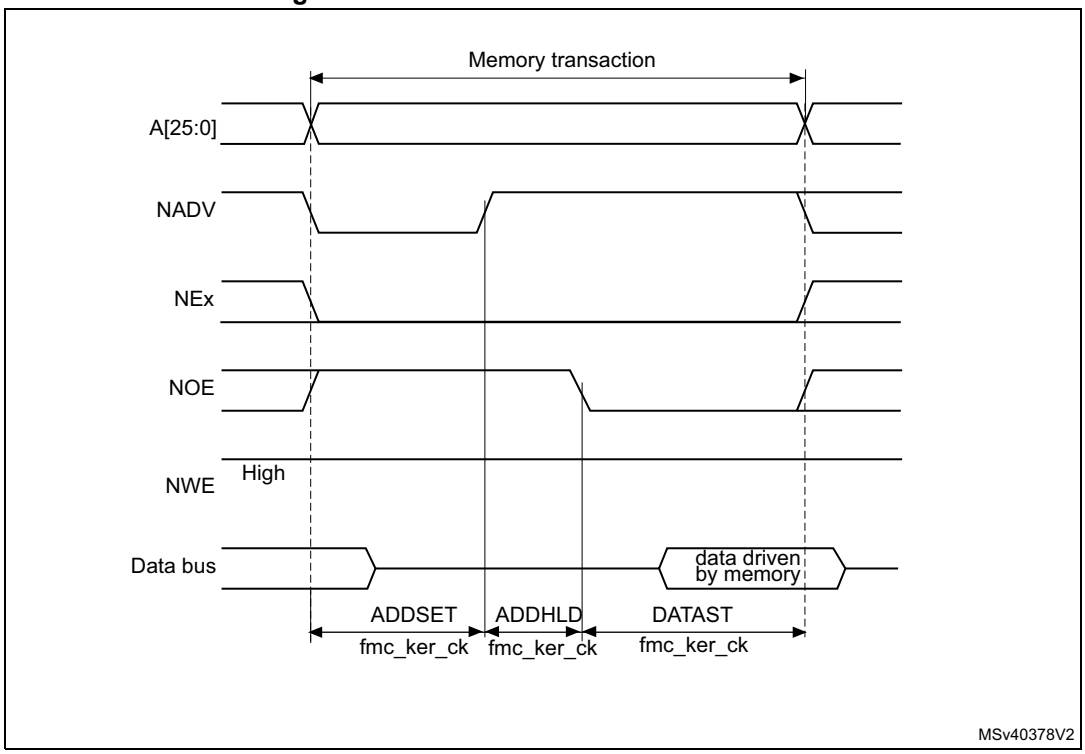
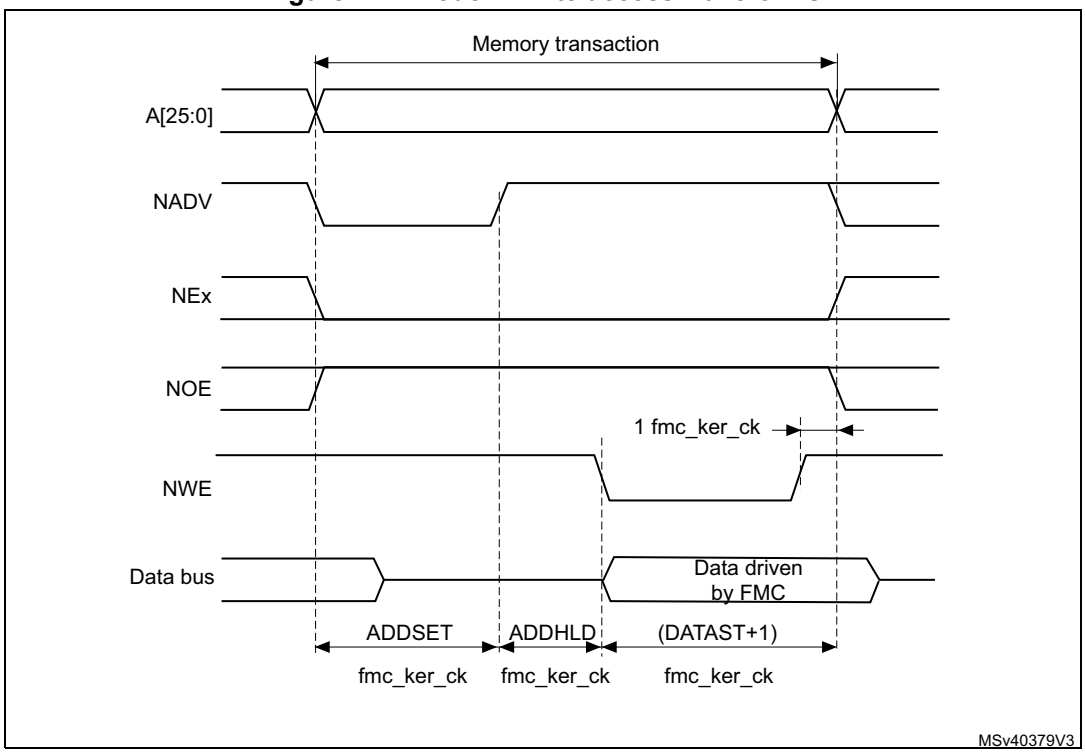


Figure 212. Mode D write access waveforms



The differences with Mode1 are the toggling of NOE that goes on toggling after NADV changes and the independent read and write timings.

Table 183. FMC_BCRx bitfields (mode D)

Bit No.	Bit name	Value to set
31	FMCEN	0x1
30:26	Reserved	0x000
25:24	BMAP	As needed
23:22	Reserved	0x000
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Set according to memory support
5:4	MWID	As needed
3:2	MTYP	As needed
1	MUXEN	0x0
0	MBKEN	0x1

Table 184. FMC_BTRx bitfields (mode D)

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x3
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.

Table 184. FMC_BTRx bitfields (mode D) (continued)

Bit number	Bit name	Value to set
7:4	ADDHLD	Duration of the middle phase of the read access (ADDHLD fmc_ker_ck cycles)
3:0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 1.

Table 185. FMC_BWTRx bitfields (mode D)

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x3
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the second access phase (DATAST + 1 fmc_ker_ck cycles) for write accesses.
7:4	ADDHLD	Duration of the middle phase of the write access (ADDHLD fmc_ker_ck cycles)
3:0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 1.

Muxed mode - multiplexed asynchronous access to NOR flash memory

Figure 213. Muxed read access waveforms

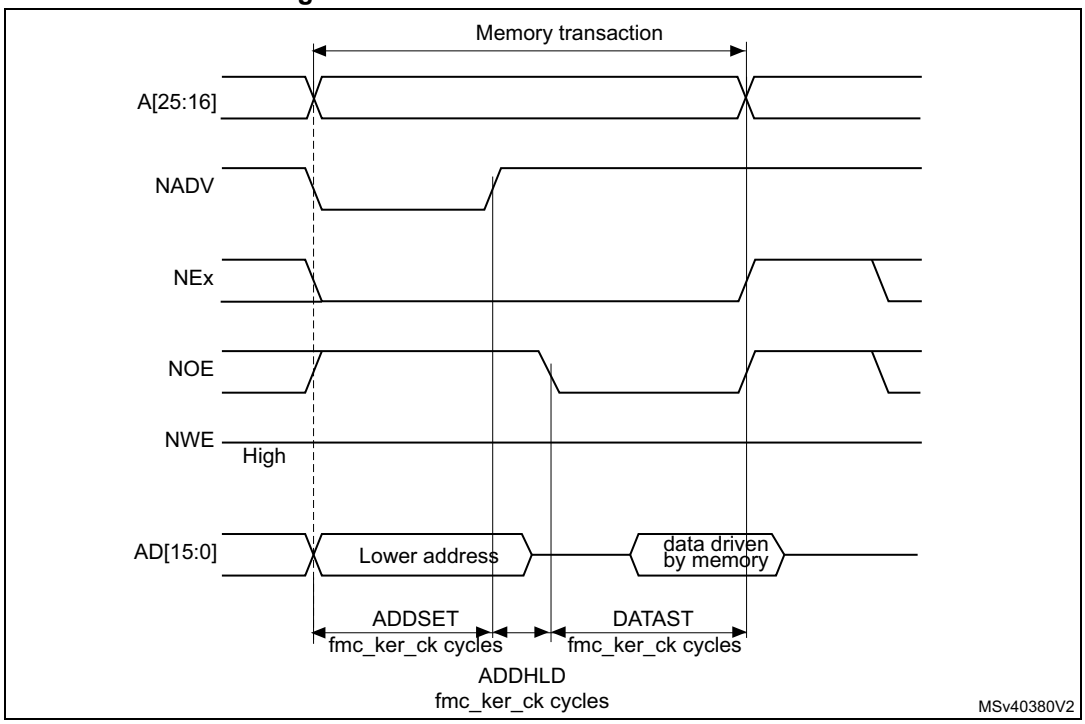
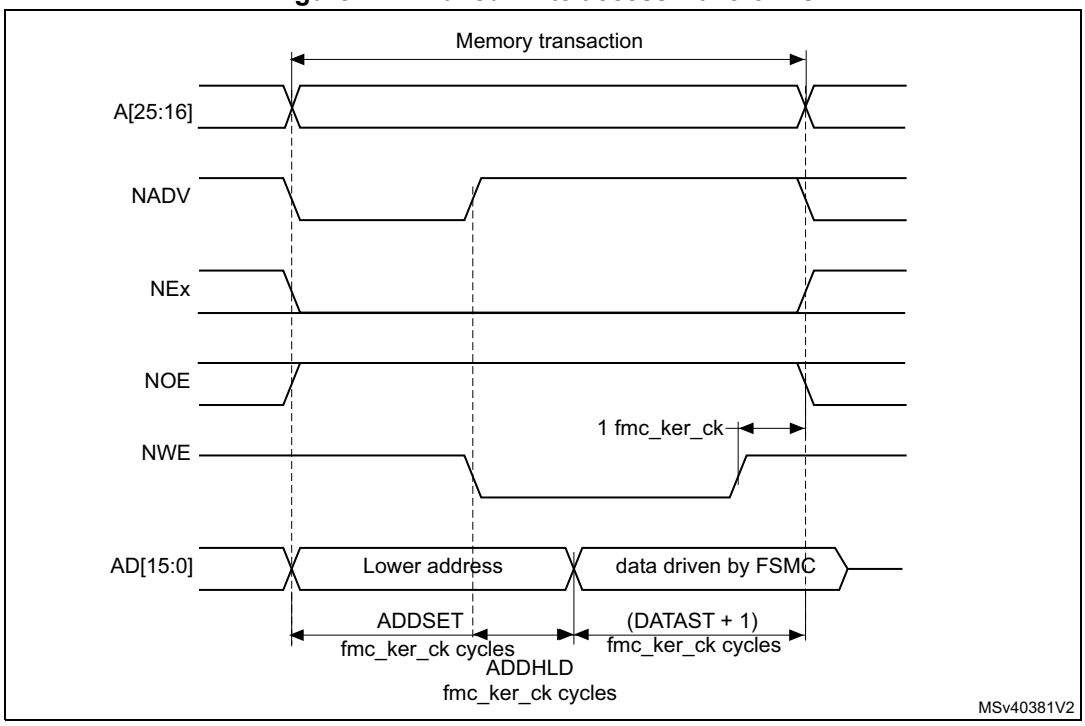


Figure 214. Muxed write access waveforms



The difference with Mode D is the drive of the lower address byte(s) on the data bus.

Table 186. FMC_BCRx bitfields (Muxed mode)

Bit No.	Bit name	Value to set
31	FMCEN	0x1
30:26	Reserved	0x000
25:24	BMAP	As needed
23:22	Reserved	0x000
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5:4	MWID	As needed
3:2	MTYP	0x2 (NOR flash memory)
1	MUXEN	0x1
0	MBKEN	0x1

Table 187. FMC_BTRx bitfields (Muxed mode)

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x0
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles for read accesses and DATAST+1 fmc_ker_ck cycles for write accesses).

Table 187. FMC_BTRx bitfields (Muxed mode) (continued)

Bit number	Bit name	Value to set
7:4	ADDHLD	Duration of the middle phase of the access (ADDHLD fmc_ker_ck cycles).
3:0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles). Minimum value for ADDSET is 1.

WAIT management in asynchronous accesses

If the asynchronous memory asserts the WAIT signal to indicate that it is not yet ready to accept or to provide data, the ASYNCWAIT bit has to be set in FMC_BCRx register.

If the WAIT signal is active (high or low depending on the WAITPOL bit), the second access phase (Data setup phase), programmed by the DATAST bits, is extended until WAIT becomes inactive. Unlike the data setup phase, the first access phases (Address setup and Address hold phases), programmed by the ADDSET and ADDHLD bits, are not WAIT sensitive and so they are not prolonged.

The data setup phase must be programmed so that WAIT can be detected 4 fmc_ker_ck cycles before the end of the memory transaction. The following cases must be considered:

1. The memory asserts the WAIT signal aligned to NOE/NWE which toggles:

$$\text{DATAST} \geq (4 \times \text{fmc_ker_ck}) + \text{max_wait_assertion_time}$$

2. The memory asserts the WAIT signal aligned to NEx (or NOE/NWE not toggling):

   if

$$\text{max_wait_assertion_time} > \text{address_phase} + \text{hold_phase}$$

   then:

$$\text{DATAST} \geq (4 \times \text{fmc_ker_ck}) + (\text{max_wait_assertion_time} - \text{address_phase} - \text{hold_phase})$$

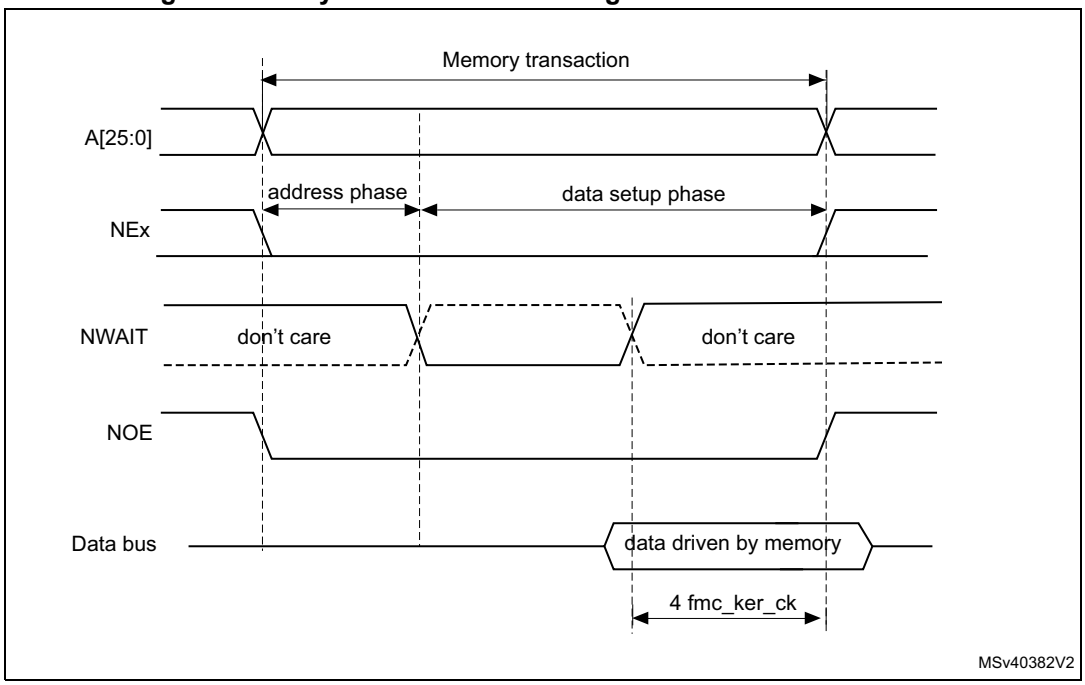
   otherwise

$$\text{DATAST} \geq (4 \times \text{fmc_ker_ck})$$

   where max_wait_assertion_time is the maximum time taken by the memory to assert the WAIT signal once NEx/NOE/NWE is low.

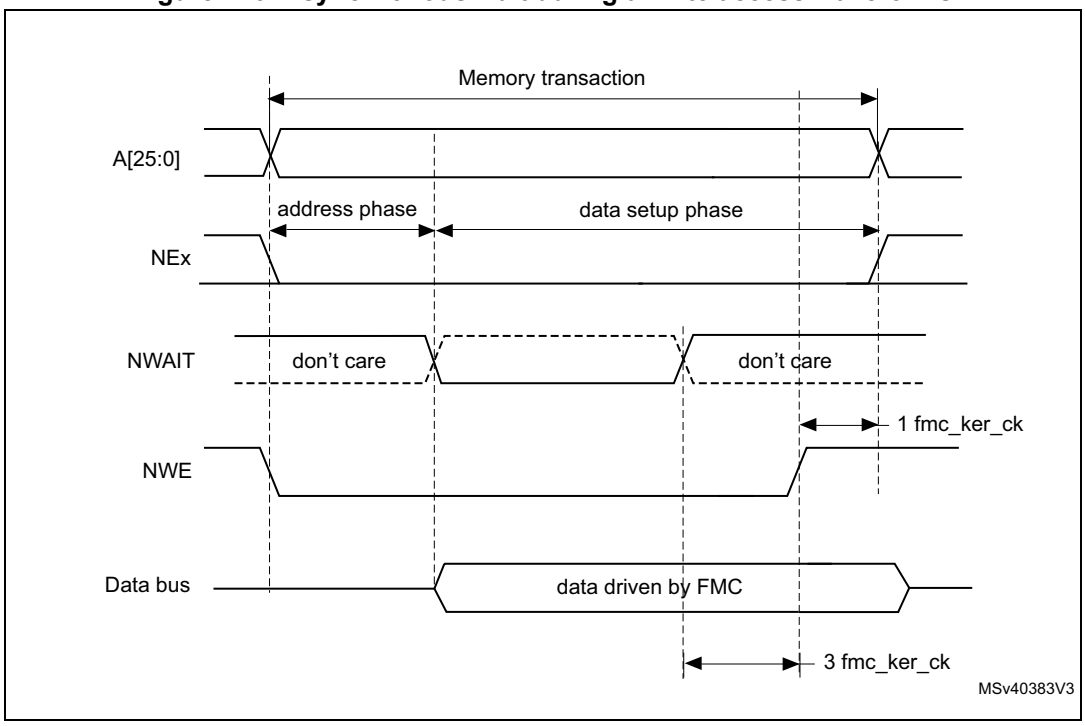
Figure 215 and *Figure 216* show the number of fmc_ker_ck clock cycles that are added to the memory access phase after WAIT is released by the asynchronous memory (independently of the above cases).

Figure 215. Asynchronous wait during a read access waveforms



1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

Figure 216. Asynchronous wait during a write access waveforms



1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

21.7.5 Synchronous transactions

The memory clock, FMC_CLK, is a sub-multiple of fmc_ker_ck. It depends on the value of CLKDIV and the MWID/ AXI data size, following the formula given below:

$$\text{FMC_CLK divider ratio} = \max(\text{CLKDIV} + 1, \text{MWID}(\text{AXI data size}))$$

Whatever MWID size: 16 or 8-bit, the FMC_CLK divider ratio is always defined by the programmed CLKDIV value.

Example:

- If CLKDIV=1, MWID = 16 bits, AXI data size=8 bits, FMC_CLK=fmc_ker_ck/2.

NOR flash memories specify a minimum time from NADV assertion to FMC_CLK high. To meet this constraint, the FMC does not issue the clock to the memory during the first internal clock cycle of the synchronous access (before NADV assertion). This guarantees that the rising edge of the memory clock occurs in the middle of the NADV low pulse.

For some PSRAM memories which must be configured to Synchronous mode, during the BCR register writing, the memory attribute space must be configured to device or strongly-ordered. Once PSRAM BCR register is configured, the memory attribute of PSRAM address space can be programmed to cacheable.

Data latency versus NOR memory latency

The data latency is the number of cycles to wait before sampling the data. The DATLAT value must be consistent with the latency value specified in the NOR flash configuration register. The FMC does not include the clock cycle when NADV is low in the data latency count.

Caution: Some NOR flash memories include the NADV Low cycle in the data latency count, so that the exact relation between the NOR flash latency and the FMC DATLAT parameter can be either:

- NOR flash latency = (DATLAT + 2) FMC_CLK clock cycles
- or NOR flash latency = (DATLAT + 3) FMC_CLK clock cycles

Some recent memories assert NWAIT during the latency phase. In such cases DATLAT can be set to its minimum value. As a result, the FMC samples the data and waits long enough to evaluate if the data are valid. Thus the FMC detects when the memory exits latency and real data are processed.

Other memories do not assert NWAIT during latency. In this case the latency must be set correctly for both the FMC and the memory, otherwise invalid data are mistaken for good data, or valid data are lost in the initial phase of the memory access.

Single-burst transfer

When the selected bank is configured in Burst mode for synchronous accesses, if for example a single-burst transaction is requested on 16-bit memories, the FMC performs a burst transaction of length 1 (if the AXI transfer is 16 bits), or length 2 (if the AXI transfer is 32 bits) and deassert the Chip Select signal when the last data is strobed.

Such transfers are not the most efficient in terms of cycles compared to asynchronous read operations. Nevertheless, a random asynchronous access would first require to re-program the memory access mode, which would altogether last longer.

Cross boundary page for Cellular RAM 1.5

Cellular RAM 1.5 does not allow burst access to cross the page boundary. The FMC allows to split automatically the burst access when the memory page size is reached by configuring the CPSIZE bits in the FMC_BCR1 register following the memory page size.

Wait management

For synchronous NOR flash memories, NWAIT is evaluated after the programmed latency period, which corresponds to (DATLAT+2) FMC_CLK clock cycles.

If NWAIT is active (low level when WAITPOL = 0, high level when WAITPOL = 1), wait states are inserted until NWAIT is inactive (high level when WAITPOL = 0, low level when WAITPOL = 1).

When NWAIT is inactive, the data is considered valid either immediately (bit WAITCFG = 1) or on the next clock edge (bit WAITCFG = 0).

During wait-state insertion via the NWAIT signal, the controller continues to send clock pulses to the memory, keeping the Chip Select and output enable signals valid. It does not consider the data as valid.

In Burst mode, there are two timing configurations for the NOR flash NWAIT signal:

- The flash memory asserts the NWAIT signal one data cycle before the wait state (default after reset).
- The flash memory asserts the NWAIT signal during the wait state

The FMC supports both NOR flash wait state configurations, for each Chip Select, thanks to the WAITCFG bit in the FMC_BCRx registers (x = 0..3).

Figure 217. Wait configuration waveforms

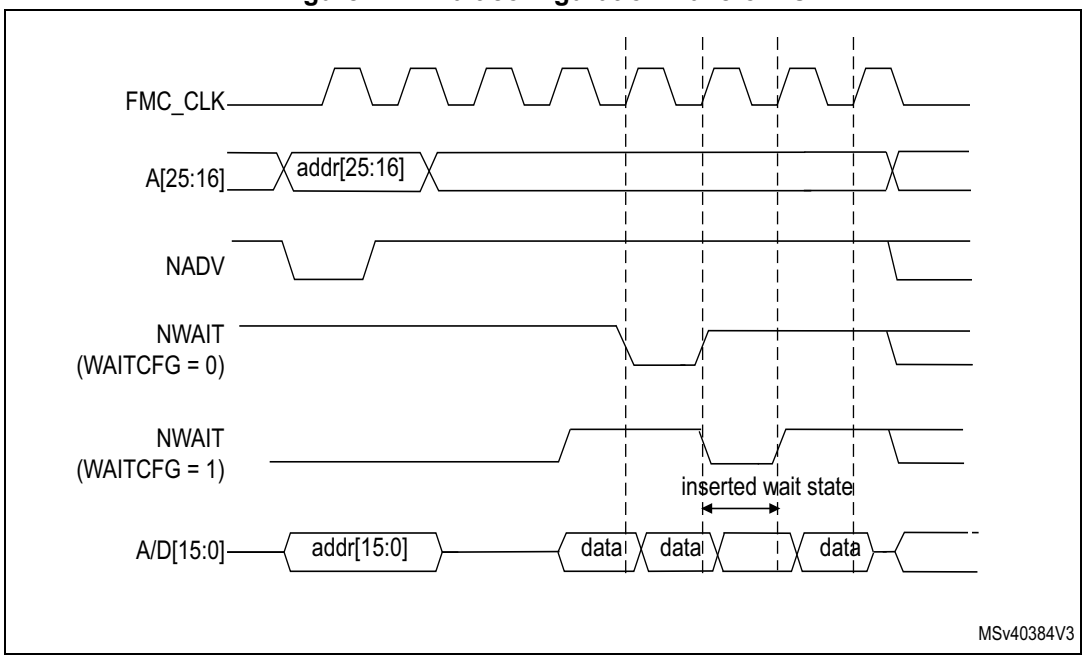
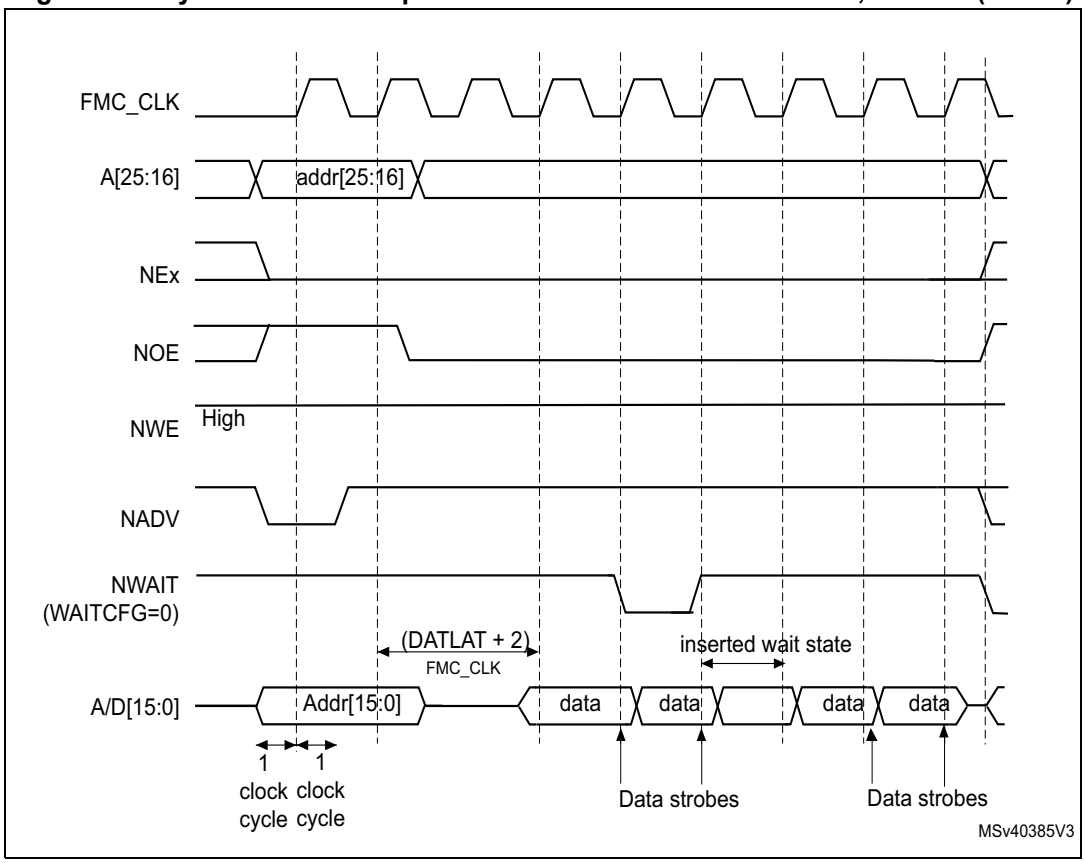


Figure 218. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)



1. Byte lane outputs (NBL are not shown; for NOR access, they are held high, and, for PSRAM (CRAM) access, they are held low.

Table 188. FMC_BCRx bitfields (Synchronous multiplexed read mode)

Bit No.	Bit name	Value to set
31	MC	0x1
30:26	Reserved	0x000
25:24	BMAP	As needed
23:22	Reserved	0x000
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	No effect on synchronous read
18:16	CPSIZE	As needed. (0x1 when using CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	to be set to 1 if the memory supports this feature, to be kept at 0 otherwise
12	WREN	no effect on synchronous read
11	WAITCFG	to be set according to memory

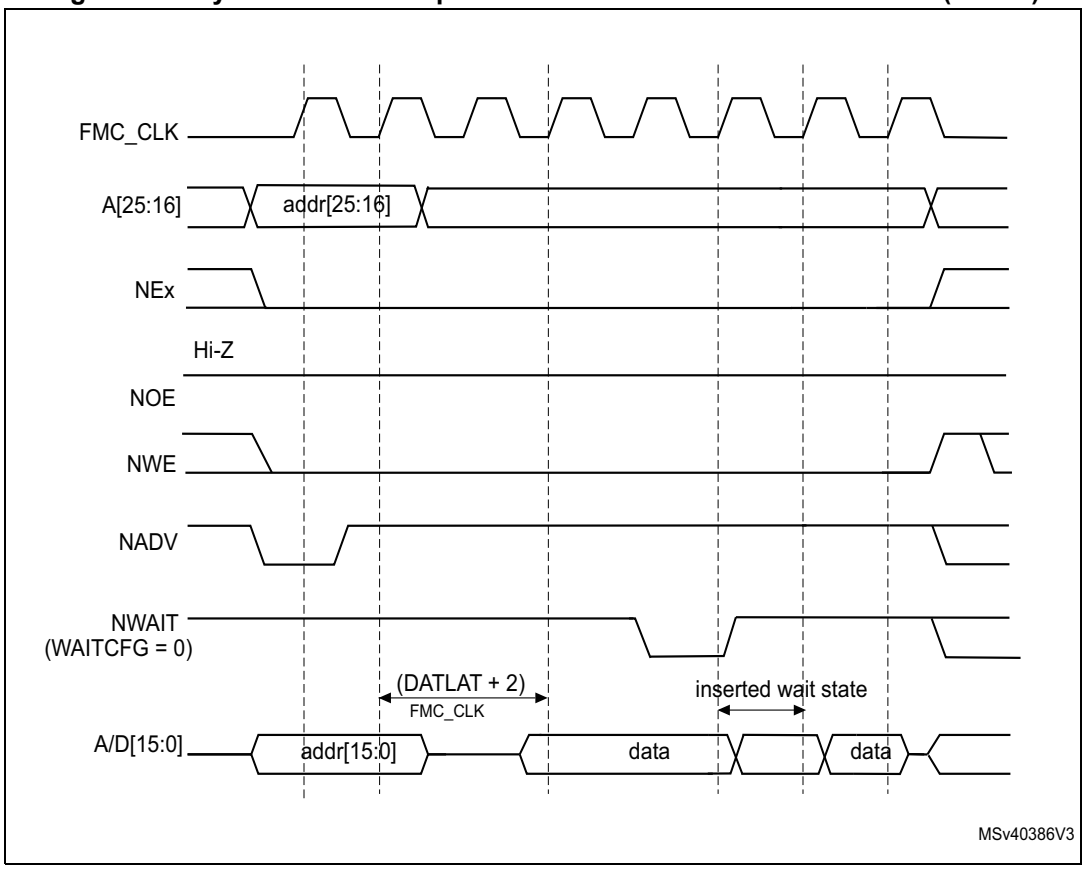
Table 188. FMC_BCRx bitfields (Synchronous multiplexed read mode) (continued)

Bit No.	Bit name	Value to set
10	Reserved	0x0
9	WAITPOL	to be set according to memory
8	BURSTEN	0x1
7	Reserved	0x1
6	FACCEN	Set according to memory support (NOR flash memory)
5:4	MWID	As needed
3:2	MTYP	0x1 or 0x2
1	MUXEN	As needed
0	MBKEN	0x1

Table 189. FMC_BTRx bitfields (Synchronous multiplexed read mode)

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x0
27:24	DATLAT	Data latency
27:24	DATLAT	Data latency
23:20	CLKDIV	0x0 to get CLK = fmc_ker_ck 0x1 to get CLK = 2 × fmc_ker_ck ..
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Don’t care
7:4	ADDHLD	Don’t care
3:0	ADDSET	Don’t care

Figure 219. Synchronous multiplexed write mode waveforms - PSRAM (CRAM)



1. The memory must issue NWAIT signal one cycle in advance, accordingly WAITCFG must be programmed to 0.
2. Byte Lane (NBL) outputs are not shown, they are held low while NEx is active.

Table 190. FMC_BCRx bitfields (Synchronous multiplexed write mode)

Bit No.	Bit name	Value to set
31	FMCEN	0x1
30:26	Reserved	0x000
25:24	BMAP	As needed
23:22	Reserved	0x000
21	WFDIS	As needed
20	CCLKEN	As needed
19	CBURSTRW	No effect on synchronous read
18:16	CPSIZE	As needed. (0x1 when using CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	to be set to 1 if the memory supports this feature, to be kept at 0 otherwise.
12	WREN	0x1

Table 190. FMC_BCRx bitfields (Synchronous multiplexed write mode) (continued)

Bit No.	Bit name	Value to set
11	WAITCFG	0x0
10	Reserved	0x0
9	WAITPOL	to be set according to memory
8	BURSTEN	no effect on synchronous write
7	Reserved	0x1
6	FACCEN	Set according to memory support
5:4	MWID	As needed
3:2	MTYP	0x1
1	MUXEN	As needed
0	MBKEN	0x1

Table 191. FMC_BTRx bitfields (Synchronous multiplexed write mode)

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x0
27:24	DATLAT	Data latency
23:20	CLKDIV	0x0 to get CLK = fmc_ker_ck 0x1 to get CLK = 2 × fmc_ker_ck
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15:8	DATAST	Don’t care
7:4	ADDHLD	Don’t care
3:0	ADDSET	Don’t care

21.7.6 NOR/PSRAM controller registers

SRAM/NOR-flash chip-select control registers for bank x (FMC_BCRx)

Address offset: 0x00 + 0x8 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 30DB, 0x0000 30D2, Block 3: 0x0000 30D2, 0x0000 30D2

This register contains the control information of each memory bank, used for SRAMs, PSRAM and NOR flash memories.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FMCEN	Res.	Res.	Res.	Res.	Res.	BMAP[1:0]		Res.	Res.	WFDIS	CCLKEN	CBURSTRW	CPSIZE[2:0]		
rw						rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
rw	rw	rw	rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 31 **FMCEN:** FMC Enable

This bit enables/disables the FMC.

0: Disable the FMC

1: Enable the FMC

Note: The FMCEN bit of the FMC_BCR2..4 registers is don't care. It is only enabled through the FMC_BCR1 register.

Bits 30:26 Reserved, must be kept at reset value.

Bits 25:24 **BMAP[1:0]:** FMC bank mapping

These bits allow different remap or swap of the FMC NOR/PSRAM and SDRAM banks (refer to *Table 156*).

00: Default mapping (refer to *Figure 201* and *Table 156*).

01: NOR/PSRAM bank and SDRAM bank 1 are swapped.

10: Reserved

11: Reserved.

Note: The BMAP bits of the FMC_BCR2..4 registers are don't care. It is only enabled through the FMC_BCR1 register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **WFDIS:** Write FIFO Disable

This bit disables the Write FIFO used by the FMC.

0: Write FIFO enabled (Default after reset)

1: Write FIFO disabled

Note: The WFDIS bit of the FMC_BCR2..4 registers is don't care. It is only enabled through the FMC_BCR1 register.

Bit 20 **CCLKEN:** Continuous Clock Enable

This bit enables the FMC_CLK clock output to external memory devices.

0: The FMC_CLK is only generated during the synchronous memory access (read/write transaction). The FMC_CLK clock ratio is specified by the programmed CLKDIV value in the FMC_BCRx register (default after reset).

1: The FMC_CLK is generated continuously during asynchronous and synchronous access. The FMC_CLK clock is activated when the CCLKEN is set.

Note: The CCLKEN bit of the FMC_BCR2..4 registers is don't care. It is only enabled through the FMC_BCR1 register. Bank 1 must be configured in Synchronous mode to generate the FMC_CLK continuous clock.

If CCLKEN bit is set, the FMC_CLK clock ratio is specified by CLKDIV value in the FMC_BTR1 register. CLKDIV in FMC_BWTR1 is don't care.

If the Synchronous mode is used and CCLKEN bit is set, the synchronous memories connected to other banks than Bank 1 are clocked by the same clock (the CLKDIV value in the FMC_BTR2..4 and FMC_BWTR2..4 registers for other banks has no effect.)

Bit 19 **CBURSTRW:** Write burst enable

For PSRAM (CRAM) operating in Burst mode, the bit enables synchronous accesses during write operations. The enable bit for synchronous read accesses is the BURSTEN bit in the FMC_BCRx register.

0: Write operations are always performed in Asynchronous mode

1: Write operations are performed in Synchronous mode.

Bits 18:16 **CPSIZE[2:0]**: CRAM Page Size

These are used for Cellular RAM 1.5 which does not allow burst access to cross the address boundaries between pages. When these bits are configured, the FMC splits automatically the burst access when the memory page size is reached (refer to memory datasheet for page size).

000: No burst split when crossing page boundary (default after reset).

001: 128 bytes

010: 256 bytes

100: 1024 bytes

Other configuration: reserved.

Bit 15 **ASYNCWAIT**: Wait signal during asynchronous transfers

This bit enables/disables the FMC to use the wait signal even during an asynchronous protocol.

0: NWAIT signal is not taken in to account when running an asynchronous protocol (default after reset)

1: NWAIT signal is taken in to account when running an asynchronous protocol

Bit 14 **EXTMOD**: Extended mode enable.

This bit enables the FMC to program the write timings for asynchronous accesses inside the FMC_BWTR register, thus resulting in different timings for read and write operations.

0: values inside FMC_BWTR register are not taken into account (default after reset)

1: values inside FMC_BWTR register are taken into account

Note: When the Extended mode is disabled, the FMC can operate in Mode1 or Mode2 as follows:

- *Mode 1 is the default mode when the SRAM/PSRAM memory type is selected (MTYP =0x0 or 0x01)*
- *Mode 2 is the default mode when the NOR memory type is selected (MTYP = 0x10).*

Bit 13 **WAITEN**: Wait enable bit

This bit enables/disables wait-state insertion via the NWAIT signal when accessing the memory in Synchronous mode.

0: NWAIT signal is disabled (its level not taken into account, no wait state inserted after the programmed flash latency period)

1: NWAIT signal is enabled (its level is taken into account after the programmed latency period to insert wait states if asserted) (default after reset)

Bit 12 **WREN**: Write enable bit

This bit indicates whether write operations are enabled/disabled in the bank by the FMC:

0: Write operations are disabled in the bank by the FMC, an AXI slave error is reported

1: Write operations are enabled for the bank by the FMC (default after reset).

Bit 11 **WAITCFG**: Wait timing configuration

The NWAIT signal indicates whether the data from the memory are valid or if a wait state must be inserted when accessing the memory in Synchronous mode. This configuration bit determines if NWAIT is asserted by the memory one clock cycle before the wait state or during the wait state:

0: NWAIT signal is active one data cycle before wait state (default after reset)

1: NWAIT signal is active during wait state (not used for PSRAM).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **WAITPOL**: Wait signal polarity bit

This bit defines the polarity of the wait signal from memory used for either in Synchronous or Asynchronous mode:

0: NWAIT active low (default after reset)

1: NWAIT active high.

Bit 8 **BURSTEN**: Burst enable bit

This bit enables/disables synchronous accesses during read operations. It is valid only for synchronous memories operating in Burst mode:

0: Burst mode disabled (default after reset). Read accesses are performed in Asynchronous mode

1: Burst mode enable. Read accesses are performed in Synchronous mode.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FACCEN**: Flash access enable

This bit enables NOR flash memory access operations.

0: Corresponding NOR flash memory access is disabled

1: Corresponding NOR flash memory access is enabled (default after reset)

Bits 5:4 **MWID[1:0]:** Memory data bus width

Defines the external memory device width, valid for all type of memories.

00: 8 bits

01: 16 bits (default after reset)

10: reserved

11: reserved

Bits 3:2 **MTYP[1:0]:** Memory type

These bits define the type of external memory attached to the corresponding memory bank:
00: SRAM (default after reset for Bank 2...4)
01: PSRAM (CRAM)
10: NOR flash/OneNAND flash (default after reset for Bank 1)
11: reserved

Bit 1 **MUXEN:** Address/data multiplexing enable bit

When this bit is set, the address and data values are multiplexed on the data bus, valid only with NOR and PSRAM memories:
0: Address/Data non-multiplexed
1: Address/Data multiplexed on databus (default after reset)

Bit 0 **MBKEN:** Memory bank enable bit

This bit enables the memory bank. After reset Bank1 is enabled, all others are disabled. Accessing a disabled bank causes an ERROR on AXI bus.
0: Corresponding memory bank is disabled
1: Corresponding memory bank is enabled

SRAM/NOR-flash chip-select timing registers for bank x (FMC_BTRx)

Address offset: 0x04 + 0x8 * (x - 1), (x = 1 to 4)

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank, used for SRAMs, PSRAM and NOR flash memories.If the EXTMOD bit is set in the FMC_BCRx register, then this register is partitioned for write and read access, that is, 2 registers are available: one to configure read accesses (this register) and one to configure write accesses (FMC_BWTRx registers).

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

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **ACCMOD[1:0]:** Access mode

These bits specify the Asynchronous access modes as shown in the timing diagrams. They are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.

00: access mode A

01: access mode B

10: access mode C

11: access mode D

Bits 27:24 **DATLAT[3:0]:** (see note below bit descriptions): Data latency for synchronous memory

For synchronous access with read/write Burst mode enabled (BURSTEN / CBURSTRW bits set), these bits define the number of memory clock cycles (+2) to issue to the memory before reading/writing the first data:

This timing parameter is not expressed in fmc_ker_ck periods, but in FMC_CLK periods.

For asynchronous access, this value is don't care.

0000: Data latency of 2 FMC_CLK clock cycles for first burst access

1111: Data latency of 17 FMC_CLK clock cycles for first burst access (default value after reset)

Bits 23:20 **CLKDIV[3:0]:** Clock divide ratio (for FMC_CLK signal)

These bits define the period of FMC_CLK clock output signal, expressed in number of fmc_ker_ck cycles:

0000: FMC_CLK period = 1 x fmc_ker_ck period

0001: FMC_CLK period = 2 × fmc_ker_ck periods

0010: FMC_CLK period = 3 × fmc_ker_ck periods

1111: FMC_CLK period = 16 × fmc_ker_ck periods (default value after reset)

In asynchronous NOR flash, SRAM or PSRAM accesses, this value is don’t care.

Note: Refer to Section 21.7.5: Synchronous transactions for FMC_CLK divider ratio formula)

Bits 19:16 **BUSTURN[3:0]:** Bus turnaround phase duration

These bits are written by software to add a delay at the end of a write-to-read (and read-to-write) transaction. This delay allows to match the minimum time between consecutive transactions (t_{EHEL} from NEx high to NEx low) and the maximum time needed by the memory to free the data bus after a read access (t_{EHQZ}). The programmed bus turnaround delay is inserted between an asynchronous read (muxed or mode D) or write transaction and any other asynchronous /synchronous read or write to or from a static bank. The bank can be the same or different in case of read, in case of write the bank can be different except for muxed or mode D.

In some cases, whatever the programmed BUSTURN values, the bus turnaround delay is fixed

as follows:

- The bus turnaround delay is not inserted between two consecutive asynchronous write transfers to the same static memory bank except for muxed and D modes.
- There is a bus turnaround delay of 1 FMC clock cycle between:
  - –Two consecutive asynchronous read transfers to the same static memory bank except for muxed and D modes.
  - –An asynchronous read to an asynchronous or synchronous write to any static bank or dynamic bank except for muxed and D modes.
  - –An asynchronous (modes 1, 2, A, B or C) read and a read from another static bank.
- There is a bus turnaround delay of 2 FMC clock cycle between:
  - –Two consecutive synchronous writes (burst or single) to the same bank.
  - –A synchronous write (burst or single) access and an asynchronous write or read transfer to or from static memory bank (the bank can be the same or different for the case of read.
  - –Two consecutive synchronous reads (burst or single) followed by any synchronous/asynchronous read or write from/to another static memory bank.
- There is a bus turnaround delay of 3 FMC clock cycle between:
  - –Two consecutive synchronous writes (burst or single) to different static bank.
  - –A synchronous write (burst or single) access and a synchronous read from the same or a different bank.

0000: BUSTURN phase duration = 0 fmc_ker_ck clock cycle added

...

1111: BUSTURN phase duration = 15 x fmc_ker_ck clock cycles added (default value after reset)

Bits 15:8 **DATAST[7:0]:** Data-phase duration

These bits are written by software to define the duration of the data phase (refer to *Figure 202* to *Figure 214*), used in asynchronous accesses:

0000 0000: Reserved

0000 0001: DATAST phase duration = 1 × fmc_ker_ck clock cycles

0000 0010: DATAST phase duration = 2 × fmc_ker_ck clock cycles

...

1111 1111: DATAST phase duration = 255 × fmc_ker_ck clock cycles (default value after reset)

For each memory type and access mode data-phase duration, refer to the respective figure (*Figure 202* to *Figure 214*).

Example: Mode1, write access, DATAST = 1: Data-phase duration = DATAST+1 = 1 x fmc_ker_ck clock cycles.

Note: In synchronous accesses, this value is don't care.

Bits 7:4 **ADDHLD[3:0]:** Address-hold phase duration

These bits are written by software to define the duration of the *address hold* phase (refer to *Figure 202* to *Figure 214*), used in mode D or multiplexed accesses:
0000: Reserved
0001: ADDHLD phase duration =1 × fmc_ker_ck clock cycle
0010: ADDHLD phase duration = 2 × fmc_ker_ck clock cycle
...
1111: ADDHLD phase duration = 15 × fmc_ker_ck clock cycles (default value after reset)
For each access mode address-hold phase duration, refer to the respective figure (*Figure 202* to *Figure 214*).

Note: In synchronous accesses, this value is not used, the address hold phase is always 1 memory clock period duration.

Bits 3:0 **ADDSET[3:0]:** Address setup phase duration

These bits are written by software to define the duration of the *address setup* phase (refer to *Figure 202* to *Figure 214*), used in SRAMs, ROMs and asynchronous NOR flash:
0000: ADDSET phase duration = 0 × fmc_ker_ck clock cycle
...
1111: ADDSET phase duration = 15 × fmc_ker_ck clock cycles (default value after reset)
For each access mode address setup phase duration, refer to the respective figure (refer to *Figure 202* to *Figure 214*).

Note: In synchronous accesses, this value is don't care.
In Muxed mode or mode D, the minimum value for ADDSET is 1.
In mode 1 and PSRAM memory, the minimum value for ADDSET is 1.

Note: PSRAMs (CRAMs) have a variable latency due to internal refresh. Therefore these memories issue the NWAIT signal during the whole latency phase to extend the latency as needed.
On PSRAMs (CRAMs) the filled DATLAT must be set to 0, so that the FMC exits its latency phase soon and starts sampling NWAIT from memory, then starts to read or write when the memory is ready.
This method can be used also with the latest generation of synchronous flash memories that issue the NWAIT signal, unlike older flash memories (check the datasheet of the specific flash memory being used).

SRAM/NOR-flash write timing registers for bank x (FMC_BWTRx)

Address offset: 0x104 + 0x8 * (x - 1), (x = 1 to 4)

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank. It is used for SRAMs, PSRAMs and NOR flash memories. When the EXTMOD bit is set in the FMC_BCRx register, then this register is active for write access.

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

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **ACCMOD[1:0]:** Access mode.

These bits specify the asynchronous access modes as shown in the next timing diagrams.These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.

00: access mode A

01: access mode B

10: access mode C

11: access mode D

Bits 27:20 Reserved, must be kept at reset value.

Bits 19:16 **BUSTURN[3:0]**: Bus turnaround phase duration

These bits are written by software to add a delay at the end of a write transaction to match the minimum time between consecutive transactions (t_{EHEL} from ENx high to ENx low):

(BUSTRUN + 1) fmc_ker_ck period ≥ $t_{EHELmin}$.

The programmed bus turnaround delay is inserted between an asynchronous write transfer and any other asynchronous /synchronous read or write transfer to or from a static bank. The bank can be the same or different in case of read, in case of write the bank can be different expect for muxed or mode D.

In some cases, whatever the programmed BUSTURN values, the bus turnaround delay is fixed as follows:

- The bus turnaround delay is not inserted between two consecutive asynchronous write transfers to the same static memory bank except for muxed and D modes.
- There is a bus turnaround delay of 2 FMC clock cycle between:
  - –Two consecutive synchronous writes (burst or single) to the same bank.
  - –A synchronous write (burst or single) transfer and an asynchronous write or read transfer to or from static memory bank.
- There is a bus turnaround delay of 3 FMC clock cycle between:
  - –Two consecutive synchronous writes (burst or single) to different static bank.
  - –A synchronous write (burst or single) transfer and a synchronous read from the same or a different bank.

0000: BUSTURN phase duration = 0 fmc_ker_ck clock cycle added

...

1111: BUSTURN phase duration = 15 fmc_ker_ck clock cycles added (default value after reset)

Bits 15:8 **DATAST[7:0]:** Data-phase duration.

These bits are written by software to define the duration of the data phase (refer to *Figure 202* to *Figure 214*), used in asynchronous SRAM, PSRAM and NOR flash memory accesses:

0000 0000: Reserved

0000 0001: DATAST phase duration = 1 × fmc_ker_ck clock cycles

0000 0010: DATAST phase duration = 2 × fmc_ker_ck clock cycles

...

1111 1111: DATAST phase duration = 255 × fmc_ker_ck clock cycles (default value after reset)

Bits 7:4 **ADDHLD[3:0]:** Address-hold phase duration.

These bits are written by software to define the duration of the *address hold* phase (refer to *Figure 202* to *Figure 214*), used in asynchronous multiplexed accesses:

0000: Reserved

0001: ADDHLD phase duration = 1 × fmc_ker_ck clock cycle

0010: ADDHLD phase duration = 2 × fmc_ker_ck clock cycle

...

1111: ADDHLD phase duration = 15 × fmc_ker_ck clock cycles (default value after reset)

Note: In synchronous NOR flash accesses, this value is not used, the address hold phase is always 1 flash clock period duration.

Bits 3:0 **ADDSET[3:0]:** Address setup phase duration.

These bits are written by software to define the duration of the *address setup* phase in fmc_ker_ck cycles (refer to *Figure 202* to *Figure 214*), used in asynchronous accesses:

0000: ADDSET phase duration = 0 × fmc_ker_ck clock cycle

...

1111: ADDSET phase duration = 15 × fmc_ker_ck clock cycles (default value after reset)

Note: In synchronous accesses, this value is not used, the address setup phase is always 1 flash clock period duration. In muxed mode, the minimum ADDSET value is 1.

21.8 NAND flash controller

The FMC generates the appropriate signal timings to drive 8- and 16-bit NAND flash memories.

The NAND bank is configured through dedicated registers (*Section 21.8.7*). The programmable memory parameters include access timings (shown in *Table 192*) and ECC configuration.

Table 192. Programmable NAND flash access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Memory setup time	Number of clock cycles (fmc_ker_ck) required to set up the address before the command assertion	Read/Write	AHB clock cycle (fmc_ker_ck)	1	255
Memory wait	Minimum duration (in fmc_ker_ck clock cycles) of the command assertion	Read/Write	AHB clock cycle (fmc_ker_ck)	2	255
Memory hold	Number of clock cycles (fmc_ker_ck) during which the address must be held (as well as the data if a write access is performed) after the command deassertion	Read/Write	AHB clock cycle (fmc_ker_ck)	1	254
Memory databus high-Z	Number of clock cycles (fmc_ker_ck) during which the data bus is kept in high-Z state after a write access has started	Write	AHB clock cycle (fmc_ker_ck)	0	254

21.8.1 External memory interface signals

The following tables list the signals that are typically used to interface NAND flash memories.

Note: The prefix “N” identifies the signals which are active low.

8-bit NAND flash memory

Table 193. 8-bit NAND flash memory

FMC pin name	I/O	Function
A[17]	O	NAND flash address latch enable (ALE) signal
A[16]	O	NAND flash command latch enable (CLE) signal
D[7:0]	I/O	8-bit multiplexed, bidirectional address/data bus
NCE	O	Chip Select
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)
NWE	O	Write enable
NWAIT/INT	I	NAND flash ready/busy input signal to the FMC

Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

16-bit NAND flash memory

Table 194. 16-bit NAND flash memory

FMC pin name	I/O	Function
A[17]	O	NAND flash address latch enable (ALE) signal
A[16]	O	NAND flash command latch enable (CLE) signal
D[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NCE	O	Chip Select
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)
NWE	O	Write enable
NWAIT/INT	I	NAND flash ready/busy input signal to the FMC

Note: Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

21.8.2 NAND flash supported memories and transactions

Table 195 shows the supported devices, access modes and transactions. Transactions not allowed (or not supported) by the NAND flash controller are shown in gray.

Table 195. Supported memories and transactions

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
NAND 8-bit	Asynchronous	R	8	8	Y	-
	Asynchronous	W	8	8	Y	-
	Asynchronous	R	16	8	Y	Split into 2 FMC accesses
	Asynchronous	W	16	8	Y	Split into 2 FMC accesses
	Asynchronous	R	32	8	Y	Split into 4 FMC accesses
	Asynchronous	W	32	8	Y	Split into 4 FMC accesses
	Asynchronous	R	32	8	Y	Split into 8 FMC accesses
	Asynchronous	W	32	8	Y	Split into 8 FMC accesses

Table 195. Supported memories and transactions (continued)

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
NAND 16-bit	Asynchronous	R	8	16	Y	-
	Asynchronous	W	8	16	N	-
	Asynchronous	R	16	16	Y	-
	Asynchronous	W	16	16	Y	-
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	32	16	Y	Split into 4 FMC accesses
	Asynchronous	W	32	16	Y	Split into 4 FMC accesses

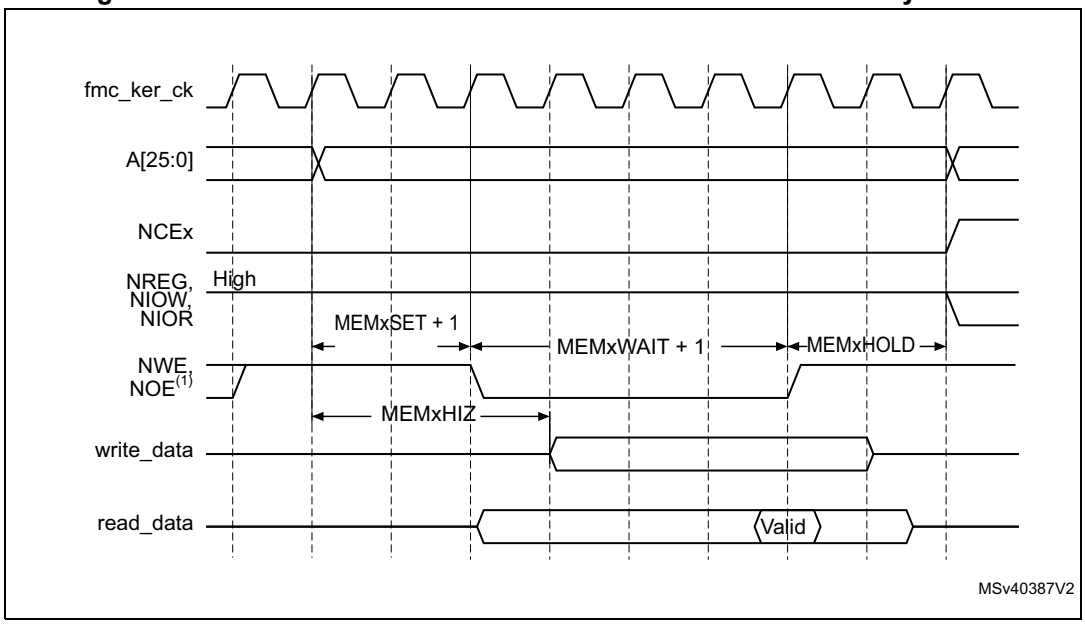
21.8.3 Timing diagrams for NAND flash memories

The NAND flash memory bank is managed through a set of registers:

- Control register: FMC_PCR
- Interrupt status register: FMC_SR
- ECC register: FMC_ECCR
- Timing register for Common memory space: FMC_PMEM
- Timing register for Attribute memory space: FMC_PATT

Each timing configuration register contains three parameters used to define the number of fmc_ker_ck cycles for the three phases of any NAND flash access, plus one parameter that defines the timing to start driving the data bus when a write access is performed. *Figure 220* shows the timing parameter definitions for common memory accesses, knowing that Attribute memory space access timings are similar.

Figure 220. NAND flash controller waveforms for common memory access



1. NOE remains high (inactive) during write accesses. NWE remains high (inactive) during read accesses.
2. For write accesses, the hold phase delay is (MEMHOLD) fmc_ker_ck cycles and for read access is (MEMHOLD + 1) fmc_ker_ck cycles.

21.8.4 NAND flash operations

The command latch enable (CLE) and address latch enable (ALE) signals of the NAND flash memory device are driven by address signals from the FMC. This means that to send a command or an address to the NAND flash memory, the CPU has to perform a write to a specific address in its memory space.

A typical page read operation from the NAND flash device requires the following steps:

1. Program and enable the corresponding memory bank by configuring the FMC_PCR and FMC_PMEM (and for some devices, FMC_PATT, see *Section 21.8.5: NAND flash prewait feature*) registers according to the characteristics of the NAND flash memory (PWID bits for the data bus width of the NAND flash memory, PWAITEN = 0 or 1 as needed, see *Section 21.6.2: NAND flash memory address mapping* for timing configuration).
2. The CPU performs a byte write to the common memory space, with data byte equal to one flash command byte (for example 0x00 for Samsung NAND flash devices). The LE input of the NAND flash memory is active during the write strobe (low pulse on NWE), thus the written byte is interpreted as a command by the NAND flash memory. Once the command is latched by the memory device, it does not need to be written again for the following page read operations.
3. The CPU can send the start address (STARTAD) for a read operation by writing four bytes (or three for smaller capacity devices), STARTAD[7:0], STARTAD[16:9], STARTAD[24:17] and finally STARTAD[25] (for 64 Mb x 8 bit NAND flash memories) in the common memory or attribute space. The ALE input of the NAND flash device is active during the write strobe (low pulse on NWE), thus the written bytes are interpreted as the start address for read operations. Using the attribute memory space makes it possible to use a different timing configuration of the FMC, which can be used

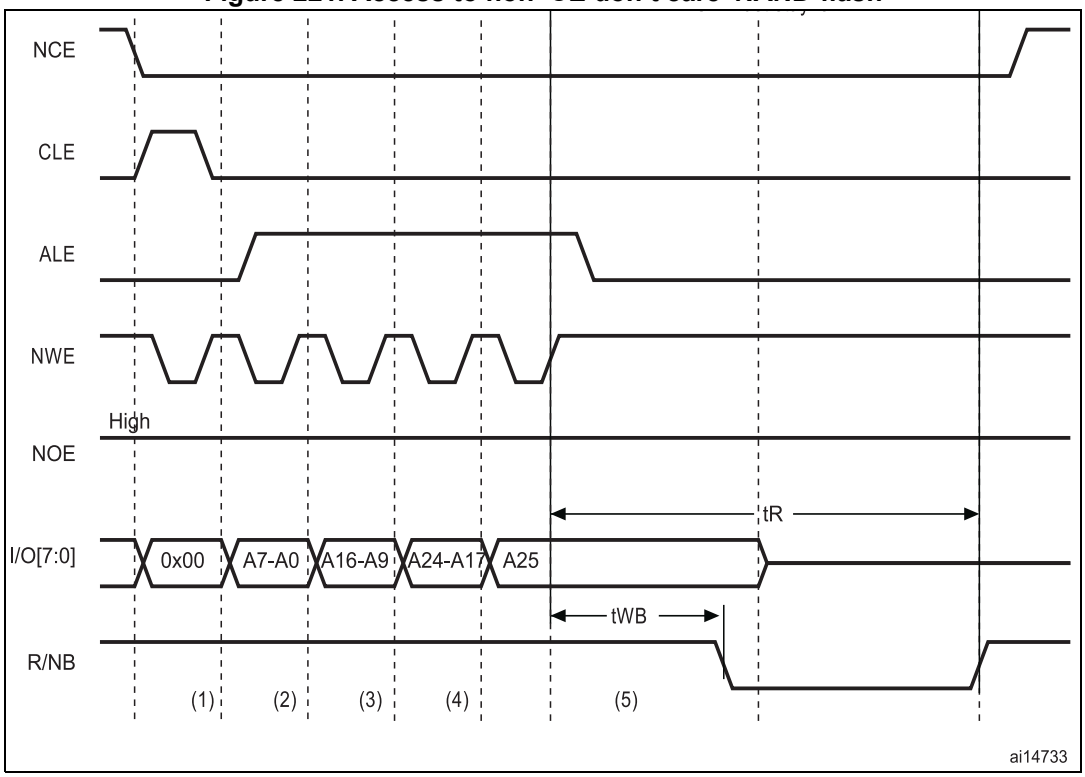
to implement the prewait functionality needed by some NAND flash memories (see details in *Section 21.8.5: NAND flash prewait feature*).

4. The controller waits for the NAND flash memory to be ready (R/NB signal high), before starting a new access to the same or another memory bank. While waiting, the controller holds the NCE signal active (low).
5. The CPU can then perform byte read operations from the common memory space to read the NAND flash page (data field + Spare field) byte by byte.
6. The next NAND flash page can be read without any CPU command or address write operation. This can be done in three different ways:
   – by simply performing the operation described in step 5
   – a new random address can be accessed by restarting the operation at step 3
   – a new command can be sent to the NAND flash device by restarting at step 2

21.8.5 NAND flash prewait feature

Some NAND flash devices require that, after writing the last part of the address, the controller waits for the R/NB signal to go low. (see *Figure 221*).

Figure 221. Access to non 'CE don't care' NAND-flash



1. CPU wrote byte 0x00 at address 0x7001 0000.
2. CPU wrote byte A7~A0 at address 0x7002 0000.
3. CPU wrote byte A16~A9 at address 0x7002 0000.
4. CPU wrote byte A24~A17 at address 0x7002 0000.
5. CPU wrote byte A25 at address 0x8802 0000: FMC performs a write access using FMC_PATT2 timing definition, where ATTHOLD ≥ 7 (providing that (7+1) × fmc_ker_ck = 112 ns > t_{WB} max). This guarantees that NCE remains low until R/NB goes low and high again (only requested for NAND flash memories where NCE is not don't care).

When this function is required, it can be performed by programming the MEMHOLD value to meet the t_{WB} timing. However, any CPU read access to NAND flash memory has a hold delay of (MEMHOLD + 1) fmc_ker_ck cycles, and any CPU write access has a hold delay of (MEMHOLD) fmc_ker_ck cycles that is inserted between the rising edge of the NWE signal and the next access.

To cope with this timing constraint, the attribute memory space can be used by programming its timing register with an ATTHOLD value that meets the t_{WB} timing, and by keeping the MEMHOLD value at its minimum value. The CPU must then use the common memory space for all NAND flash read and write accesses, except when writing the last address byte to the NAND flash device, where the CPU must write to the attribute memory space.

21.8.6 Computation of the error correction code (ECC) in NAND flash memory

The FMC includes an error correction code computation hardware block. It reduces the host CPU workload when processing the ECC by software.The ECC block is associated with NAND bank.

The ECC algorithm implemented in the FMC can perform 1-bit error correction and 2-bit error detection per 256, 512, 1 024, 2 048, 4 096 or 8 192 bytes read or written from/to the NAND flash memory. It is based on the Hamming coding algorithm and consists in calculating the row and column parity.

The ECC modules monitor the NAND flash data bus and read/write signals (NCE and NWE) each time the NAND flash memory bank is active.

The ECC operates as follows:

- When accessing NAND flash bank, the data present on the D[15:0] bus is latched and used for ECC computation.
- When accessing any other address in NAND flash memory, the ECC logic is idle, and does not perform any operation. As a result, write operations to define commands or addresses to the NAND flash memory are not taken into account for ECC computation.

Once the desired number of bytes has been read/written from/to the NAND flash memory by the host CPU, the FMC_ECCR registers must be read to retrieve the computed value. Once read, they should be cleared by resetting the ECCEN bit to ‘0’. To compute a new data block, the ECCEN bit must be set to one in the FMC_PCR registers.

Execute below the sequence to perform an ECC computation:

1. Enable the ECCEN bit in the FMC_PCR register.
2. Write data to the NAND flash memory page. While the NAND page is written, the ECC block computes the ECC value.
3. Wait until the ECC code is ready (FIFO empty).
4. Read the ECC value available in the FMC_ECCR register and store it in a variable.
5. Clear the ECCEN bit and then enable it in the FMC_PCR register before reading back the written data from the NAND page. While the NAND page is read, the ECC block computes the ECC value.
6. Read the new ECC value available in the FMC_ECCR register.
7. If the two ECC values are the same, no correction is required, otherwise there is an ECC error and the software correction routine returns information on whether the error can be corrected or not.

21.8.7 NAND flash controller registers

NAND flash control registers (FMC_PCR)

Address offset: 0x80

Reset value: 0x0000 0018

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCPS[2:0]			TAR3
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAR[2:0]			TCLR[3:0]				Res.	Res.	ECCEN	PWID[1:0]		Res.	PBKEN	PWAITEN	Res.
rw	rw	rw	rw	rw	rw	rw			rw	rw	rw		rw	rw	

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:17 **ECCPS[2:0]:** ECC page size.
These bits define the page size for the extended ECC:
000: 256 bytes
001: 512 bytes
010: 1024 bytes
011: 2048 bytes
100: 4096 bytes
101: 8192 bytes

Bits 16:13 **TAR[3:0]:** ALE to RE delay.
These bits set time from ALE low to RE low in number of fmc_ker_ck clock cycles.
Time is: t_ar = (TAR + SET + 2) × $t_{fmc_ker_ck}$ where $t_{fmc_ker_ck}$ is the FMC clock period
0000: 1 x fmc_ker_ck cycle (default)
1111: 16 x fmc_ker_ck cycles
Note: Set is MEMSET or ATTSET according to the addressed space.

Bits 12:9 **TCLR[3:0]:** CLE to RE delay.
These bits set time from CLE low to RE low in number of fmc_ker_ck clock cycles. The time is give by the following formula:
t_clr = (TCLR + SET + 2) × $t_{fmc_ker_ck}$ where $t_{fmc_ker_ck}$ is the fmc_ker_ck clock period
0000: 1 x fmc_ker_ck cycle (default)
1111: 16 x fmc_ker_ck cycles
Note: Set is MEMSET or ATTSET according to the addressed space.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **ECCEN:** ECC computation logic enable bit
0: ECC logic is disabled and reset (default after reset),
1: ECC logic is enabled.

Bits 5:4 **PWID[1:0]:** Data bus width.
These bits define the external memory device width.
00: 8 bits
01: 16 bits (default after reset).
10: reserved.
11: reserved.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **PBKEN:** NAND flash memory bank enable bit.

This bit enables the memory bank. Accessing a disabled memory bank causes an ERROR on AXI bus

0: Corresponding memory bank is disabled (default after reset)

1: Corresponding memory bank is enabled

Bit 1 **PWAITEN:** Wait feature enable bit.

This bit enables the Wait feature for the NAND flash memory bank:

0: disabled

1: enabled

Bit 0 Reserved, must be kept at reset value.

FIFO status and interrupt register (FMC_SR)

Address offset: 0x84

Reset value: 0x0000 0040

This register contains information about the FIFO status and interrupt. The FMC features a FIFO that is used when writing to memories to transfer up to 16 words of data.

This is used to quickly write to the FIFO and free the AXI bus for transactions to peripherals other than the FMC, while the FMC is draining its FIFO into the memory. One of these register bits indicates the status of the FIFO, for ECC purposes.

The ECC is calculated while the data are written to the memory. To read the correct ECC, the software must consequently wait until the FIFO is empty.

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **FEMPT:** FIFO empty.

Read-only bit that provides the status of the FIFO

0: FIFO not empty

1: FIFO empty

Bit 5 **IFEN:** Interrupt falling edge detection enable bit

0: Interrupt falling edge detection request disabled

1: Interrupt falling edge detection request enabled

Bit 4 **ILEN:** Interrupt high-level detection enable bit

0: Interrupt high-level detection request disabled

1: Interrupt high-level detection request enabled

Bit 3 **IREN:** Interrupt rising edge detection enable bit

0: Interrupt rising edge detection request disabled

1: Interrupt rising edge detection request enabled

Bit 2 **IFS:** Interrupt falling edge status
The flag is set by hardware and reset by software.
0: No interrupt falling edge occurred
1: Interrupt falling edge occurred
Note: If this bit is written by software to 1 it is set.

Bit 1 **ILS:** Interrupt high-level status
The flag is set by hardware and reset by software.
0: No Interrupt high-level occurred
1: Interrupt high-level occurred

Bit 0 **IRS:** Interrupt rising edge status
The flag is set by hardware and reset by software.
0: No interrupt rising edge occurred
1: Interrupt rising edge occurred
Note: If this bit is written by software to 1 it is set.

Common memory space timing register (FMC_PMEM)

Address offset: 0x88

Reset value: 0xFCFC FCFC

The FMC_PMEM read/write register contains the timing information for NAND flash memory bank. This information is used to access either the common memory space of the NAND flash for command, address write access and data read/write access.

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

Bits 31:24 **MEMHIZ[7:0]:** Common memory x data bus Hi-Z time
These bits define the number of fmc_ker_ck clock cycles during which the data bus is kept Hi-Z after the start of a NAND flash write access to common memory space. This is only valid for write transactions:
0000 0000: 0 x fmc_ker_ck cycle
1111 1110: 254 x fmc_ker_ck cycles
1111 1111: reserved.

Bits 23:16 **MEMHOLD[7:0]:** Common memory hold time
These bits define the number of fmc_ker_ck clock cycles for write accesses and fmc_ker_ck+1 clock cycles for read accesses during which the address is held (and data for write accesses) after the command is deasserted (NWE, NOE), for NAND flash read or write access to common memory space:
0000 0000: reserved.
0000 0001: 1 fmc_ker_ck cycle for write access / 3 fmc_ker_ck cycle for read access
1111 1110: 254 fmc_ker_ck cycles for write access / 257 fmc_ker_ck cycles for read access
1111 1111: reserved.

Bits 15:8 **MEMWAIT[7:0]:** Common memory wait time

These bits define the minimum number of fmc_ker_ck (+1) clock cycles to assert the command (NWE, NOE), for NAND flash read or write access to common memory space. The duration of command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of fmc_ker_ck:
0000 0000: reserved
0000 0001: x fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)
1111 1110: 255 x fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)
1111 1111: reserved.

Bits 7:0 **MEMSET[7:0]:** Common memory x setup time

These bits define the number of fmc_ker_ck (+1) clock cycles to set up the address before the command assertion (NWE, NOE), for NAND flash read or write access to common memory space:
0000 0000: fmc_ker_ck cycles
1111 1110: 255 x fmc_ker_ck cycles
1111 1111: reserved

Attribute memory space timing registers (FMC_PATT)

Address offset: 0x8C

Reset value: 0xFCFC FCFC

The FMC_PATT read/write register contains the timing information for NAND flash memory bank. It is used for 8-bit accesses to the attribute memory space of the NAND flash for the last address write access if the timing must differ from that of previous accesses (for Ready/Busy management, refer to *Section 21.8.5: NAND flash prewait feature*).

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

Bits 31:24 **ATTHIZ[7:0]:** Attribute memory data bus Hi-Z time

These bits define the number of fmc_ker_ck clock cycles during which the data bus is kept in Hi-Z after the start of a NAND flash write access to attribute memory space on socket. Only valid for writ transaction:
0000 0000: 0 x fmc_ker_ck cycle
1111 1110: 254 x fmc_ker_ck cycles
1111 1111: reserved.

Bits 23:16 **ATTHOLD[7:0]:** Attribute memory hold time

These bits define the number of fmc_ker_ck clock cycles during which the address is held (and data for write access) after the command deassertion (NWE, NOE), for NAND flash read or write access to attribute memory space:
0000 0000: reserved
0000 0001: 1 x fmc_ker_ck cycle
1111 1110: 254 x fmc_ker_ck cycles
1111 1111: reserved.

Bits 15:8 **ATTWAIT[7:0]:** Attribute memory wait time

These bits define the minimum number of x fmc_ker_ck (+1) clock cycles to assert the command (NWE, NOE), for NAND flash read or write access to attribute memory space. The duration for command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of fmc_ker_ck:

0000 0000: reserved

0000 0001: 2 x fmc_ker_ck cycles (+ wait cycle introduced by deassertion of NWAIT)

1111 1110: 255 x fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)

1111 1111: reserved.

Bits 7:0 **ATTSET[7:0]:** Attribute memory setup time

These bits define the number of fmc_ker_ck (+1) clock cycles to set up address before the command assertion (NWE, NOE), for NAND flash read or write access to attribute memory space:

0000 0000: 1 x fmc_ker_ck cycle

1111 1110: 255 x fmc_ker_ck cycles

1111 1111: reserved.

ECC result registers (FMC_ECCR)

Address offset: 0x94

Reset value: 0x0000 0000

This register contain the current error correction code value computed by the ECC computation modules of the FMC NAND controller. When the CPU reads/writes the data from a NAND flash memory page at the correct address (refer to *Section 21.8.6: Computation of the error correction code (ECC) in NAND flash memory*), the data read/written from/to the NAND flash memory are processed automatically by the ECC computation module. When X bytes have been read (according to the ECCPS field in the FMC_PCR registers), the CPU must read the computed ECC value from the FMC_ECC registers. It then verifies if these computed parity data are the same as the parity value recorded in the spare area, to determine whether a page is valid, and, to correct it otherwise. The FMC_ECCR register should be cleared after being read by setting the ECCEN bit to ‘0’. To compute a new data block, the ECCEN bit must be set to ’1’.

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

Bits 31:0 **ECC[31:0]:** ECC result

This field contains the value computed by the ECC computation logic. *Table 196* describes the contents of these bitfields.

Table 196. ECC result relevant bits

ECCPS[2:0]	Page size in bytes	ECC bits
000	256	ECC[21:0]
001	512	ECC[23:0]
010	1024	ECC[25:0]
011	2048	ECC[27:0]
100	4096	ECC[29:0]
101	8192	ECC[31:0]

21.9 SDRAM controller

21.9.1 SDRAM controller main features

The main features of the SDRAM controller are the following:

- Two SDRAM banks with independent configuration
- 8-bit, 16-bit data bus width
- 13-bits Address Row, 11-bits Address Column, 4 internal banks: 4x16Mx16bit (128 MB), 4x16Mx8bit (64 MB)
- Word, half-word, byte access
- SDRAM clock can be fmc_ker_ck/2 or fmc_ker_ck/3
- Automatic row and bank boundary management
- Multibank ping-pong access
- Programmable timing parameters
- Automatic Refresh operation with programmable Refresh rate
- Self-refresh mode
- Power-down mode
- SDRAM power-up initialization by software
- CAS latency of 1,2,3
- Cacheable Read FIFO with depth of 6 lines x32-bit (6 x14-bit address tag)

21.9.2 SDRAM External memory interface signals

At startup, the SDRAM I/O pins used to interface the FMC SDRAM controller with the external SDRAM devices must configured by the user application. The SDRAM controller I/O pins which are not used by the application, can be used for other purposes.

Table 197. SDRAM signals

SDRAM signal	I/O type	Description	Alternate function
SDCLK	O	SDRAM clock	-
SDCKE[1:0]	O	SDCKE0: SDRAM Bank 1 Clock Enable SDCKE1: SDRAM Bank 2 Clock Enable	-
SDNE[1:0]	O	SDNE0: SDRAM Bank 1 Chip Enable SDNE1: SDRAM Bank 2 Chip Enable	-
A[12:0]	O	Address	FMC_A[12:0]
D[15:0]	I/O	Bidirectional data bus	FMC_D[15:0]
BA[1:0]	O	Bank Address	FMC_A[15:14]
NRAS	O	Row Address Strobe	-
NCAS	O	Column Address Strobe	-
SDNWE	O	Write Enable	-
NBL[1:0]	O	Output Byte Mask for write accesses (memory signal name: DQM[1:0]	FMC_NBL[1:0]

21.9.3 SDRAM controller functional description

All SDRAM controller outputs (signals, address and data) change on the falling edge of the memory clock (FMC_SDCLK).

SDRAM initialization

The initialization sequence is managed by software. If the two banks are used, the initialization sequence must be generated simultaneously to Bank 1and Bank 2 by setting the Target Bank bits CTB1 and CTB2 in the FMC_SDCMR register:

1. Program the memory device features into the FMC_SDCRx register. The SDRAM clock frequency, RBURST and RPIPE must be programmed in the FMC_SDCR1 register.
2. Program the memory device timing into the FMC_SDTRx register. The TRP and TRC timings must be programmed in the FMC_SDTR1 register.
3. Set MODE bits to ‘001’ and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to start delivering the clock to the memory (SDCKE is driven high).
4. Wait during the prescribed delay period. Typical delay is around 100 µs (refer to the SDRAM datasheet for the required delay after power-up).
5. Set MODE bits to ‘010’ and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to issue a “Precharge All” command.
6. Set MODE bits to ‘011’, and configure the Target Bank bits (CTB1 and/or CTB2) as well as the number of consecutive Auto-refresh commands (NRFS) in the FMC_SDCMR register. Refer to the SDRAM datasheet for the number of Auto-refresh commands that should be issued. Typical number is 8.
7. Configure the MRD field, set the MODE bits to ‘100’, and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to issue a “Load Mode Register” command and program the SDRAM device. In particular the Burst Length (BL) has to be set to ‘1’) and the CAS latency has to be selected. If the Mode Register is not the same for both SDRAM banks, this step has to be repeated twice, once for each bank and the Target Bank bits set accordingly. For mobile SDRAM devices, the MRD field is also used to configure the extended mode register while issuing the Load Mode Register”
8. Program the refresh rate in the FMC_SDRTR register

   The refresh rate corresponds to the delay between refresh cycles. Its value must be adapted to SDRAM devices.

At this stage the SDRAM device is ready to accept commands. If a system reset occurs during an ongoing SDRAM access, the data bus might still be driven by the SDRAM device. Therefor the SDRAM device must be first reinitialized after reset before issuing any new access by the NOR flash/PSRAM/SRAM or NAND flash controller.

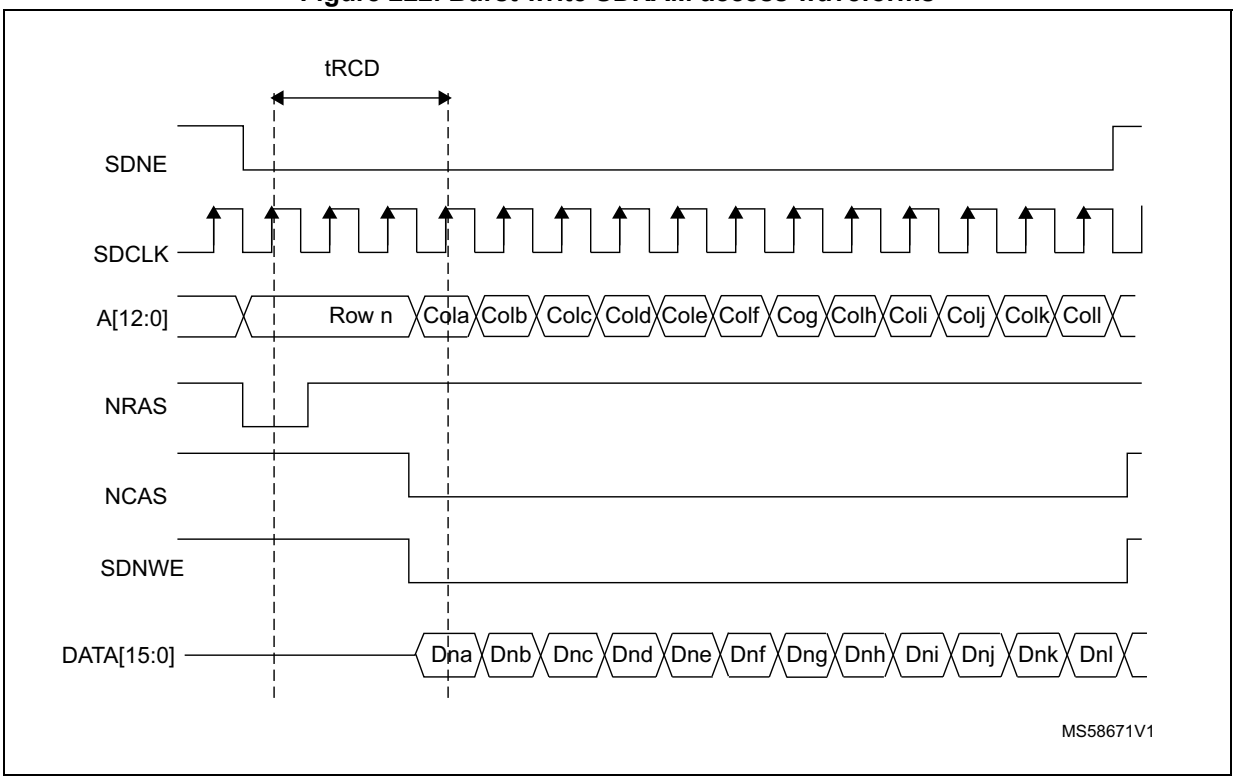
Note: If two SDRAM devices are connected to the FMC, all the accesses performed at the same time to both devices by the Command Mode register (Load Mode Register command) are issued using the timing parameters configured for SDRAM Bank 1 (TMRD andTRAS timings) in the FMC_SDTR1 register.

SDRAM controller write cycle

The SDRAM controller accepts single and burst write requests and translates them into single memory accesses. In both cases, the SDRAM controller keeps track of the active row for each bank to be able to perform consecutive write accesses to different banks (Multibank ping-pong access).

Before performing any write access, the SDRAM bank write protection must be disabled by clearing the WP bit in the FMC_SDCRx register.

Figure 222. Burst write SDRAM access waveforms



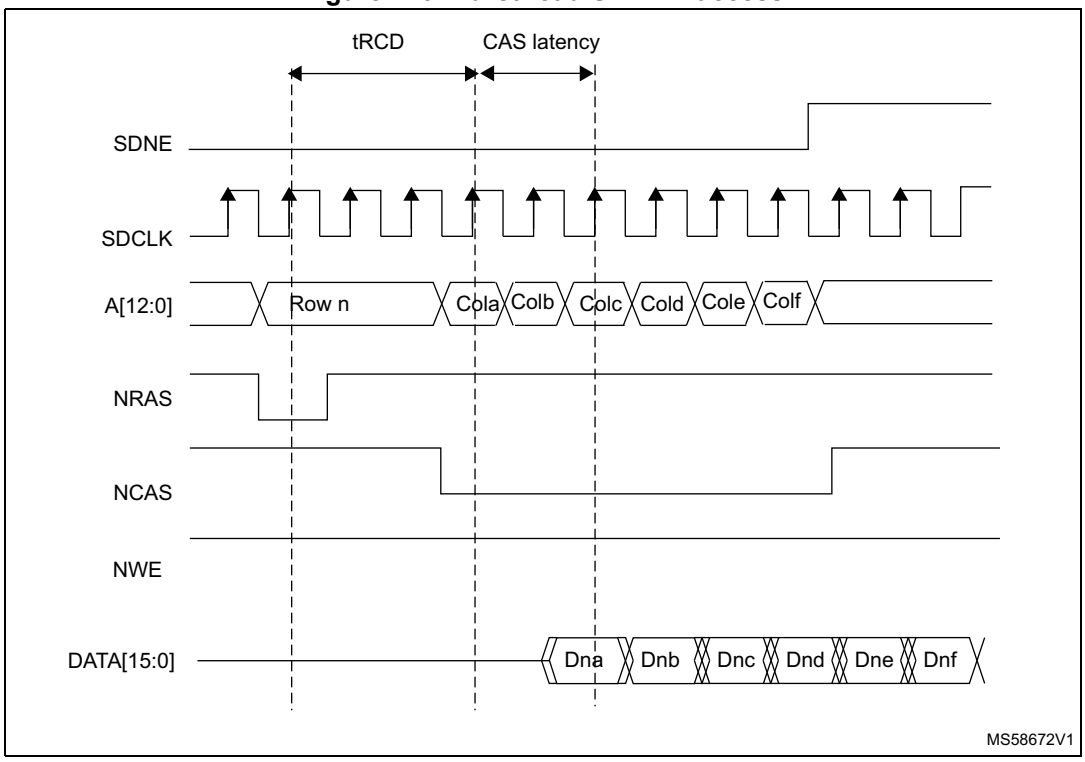
The SDRAM controller always checks the next access.

- If the next access is in the same row or in another active row, the write operation is carried out,
- if the next access targets another row (not active), the SDRAM controller generates a precharge command, activates the new row and initiates a write command.

SDRAM controller read cycle

The SDRAM controller accepts single and burst read requests and translates them into single memory accesses. In both cases, the SDRAM controller keeps track of the active row in each bank to be able to perform consecutive read accesses in different banks (Multibank ping-pong access).

Figure 223. Burst read SDRAM access



The FMC SDRAM controller features a Cacheable read FIFO (6 lines x 32 bits). It is used to store data read in advance during the CAS latency period (up to 3 memory clock cycles, programmed FMC_SDCRx) and during the RPIPE delay when set to 2xfmc_ker_ck clock cycles as configured in FMC_SDCR1) following this formula: CAS Latency + 1 + (RPIPE DIV2). The RBURST bit must be set in the FMC_SDCR1 register to anticipate the next read access.

Examples:

- CAS=3, RPIPE= 2xfmc_ker_ck. In this case, 5 data (not committed) are stored in the FIFO (4 data during CAS latency and 1 data during RPIPE delay)
- CAS=3, RPIPE= 1xfmc_ker_ck. In this case, 4 data (not committed) are stored in the FIFO (4 data during CAS latency)

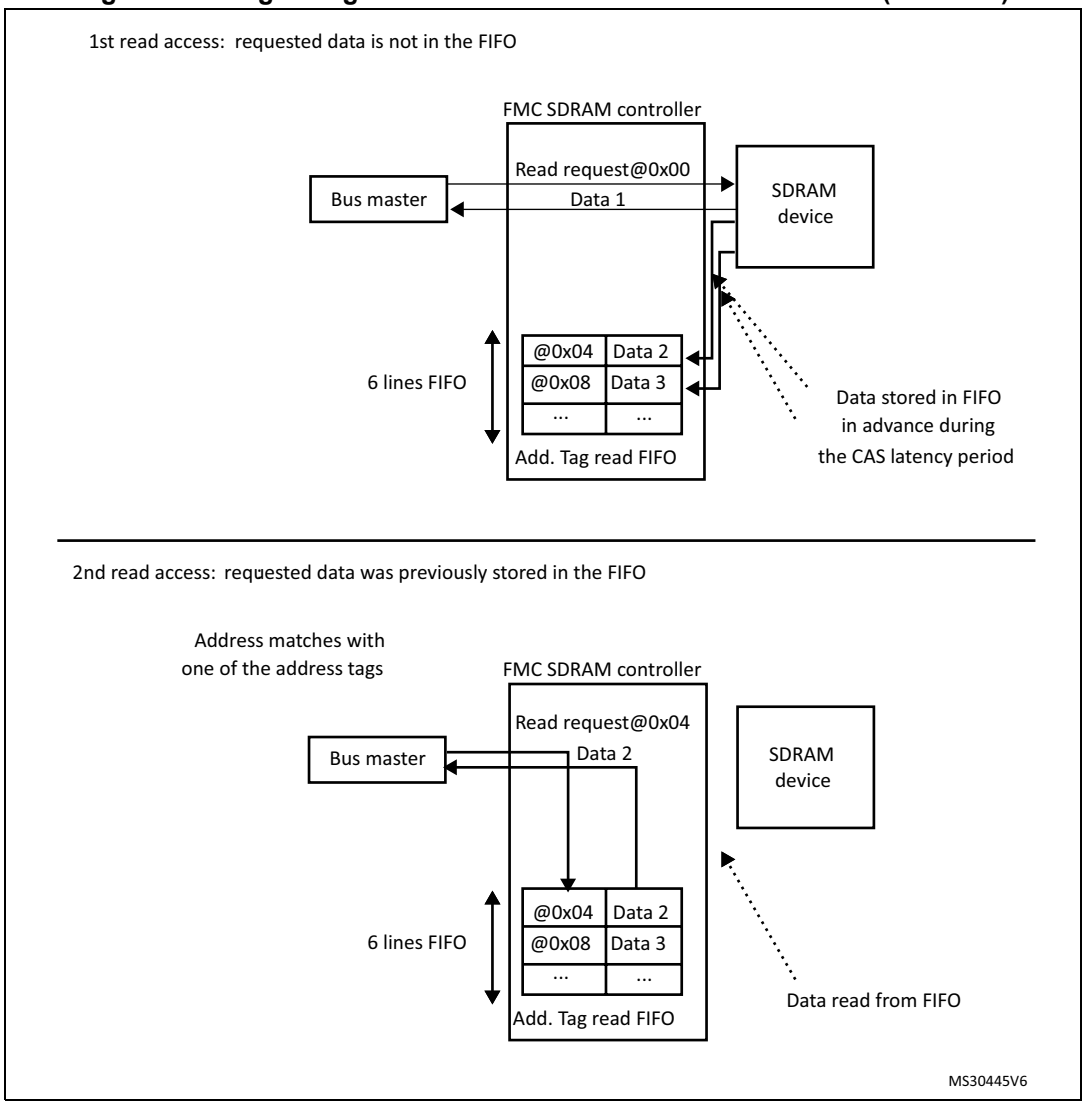
The read FIFO features a 14-bit address tag to each line to identify its content: 11 bits for the column address, 2 bits to select the internal bank and the active row, and 1 bit to select the SDRAM device

When the end of the row is reached in advance during an burst read transaction, the data read in advance (not committed) are not stored in the read FIFO. For single read access, data are correctly stored in the FIFO.

Each time a read request occurs, the SDRAM controller checks:

- If the address matches one of the address tags, data are directly read from the FIFO and the corresponding address tag/ line content is cleared and the remaining data in the FIFO are compacted to avoid empty lines.
- Otherwise, a new read command is issued to the memory and the FIFO is updated with new data. If the FIFO is full, the older data are lost.

Figure 224. Logic diagram of Read access with RBURST bit set (RPIPE=0)



During a write access or a Precharge command, the read FIFO is flushed and ready to be filled with new data.

After the first read request, if the current access was not performed to a row boundary, the SDRAM controller anticipates the next read access during the CAS latency period and the RPIPE delay (if configured). This is done by incrementing the memory address. The following condition must be met:

- RBURST control bit should be set to ‘1’ in the FMC_SDCR1 register.

The address management depends on the next AXI request:

- Next request is sequential (Burst access)

  In this case, the SDRAM controller increments the address.
- Next request is not sequential
  - If the new read request targets the same row or another active row, the new address is passed to the memory and the master is stalled for the CAS latency period, waiting for the new data from memory.
  - If the new read request does not target an active row, the SDRAM controller generates a Precharge command, activates the new row, and initiates a read command.

If the RBURST is reset, the read FIFO is not used.

Row and bank boundary management

When a read or write access crosses a row boundary, if the next read or write access is sequential and the current access was performed to a row boundary, the SDRAM controller executes the following operations:

1. Precharge of the active row,
2. Activation of the new row
3. Start of a read/write command.

At a row boundary, the automatic activation of the next row is supported for all columns and data bus width configurations.

If necessary, the SDRAM controller inserts additional clock cycles between the following commands:

- Between Precharge and Active commands to match TRP parameter (only if the next access is in a different row in the same bank),
- Between Active and Read commands to match the TRCD parameter.

These parameters are defined into the FMC_SDTRx register.

Refer to *Figure 222* and *Figure 223* for read and burst write access crossing a row boundary.

Figure 225. Read access crossing row boundary

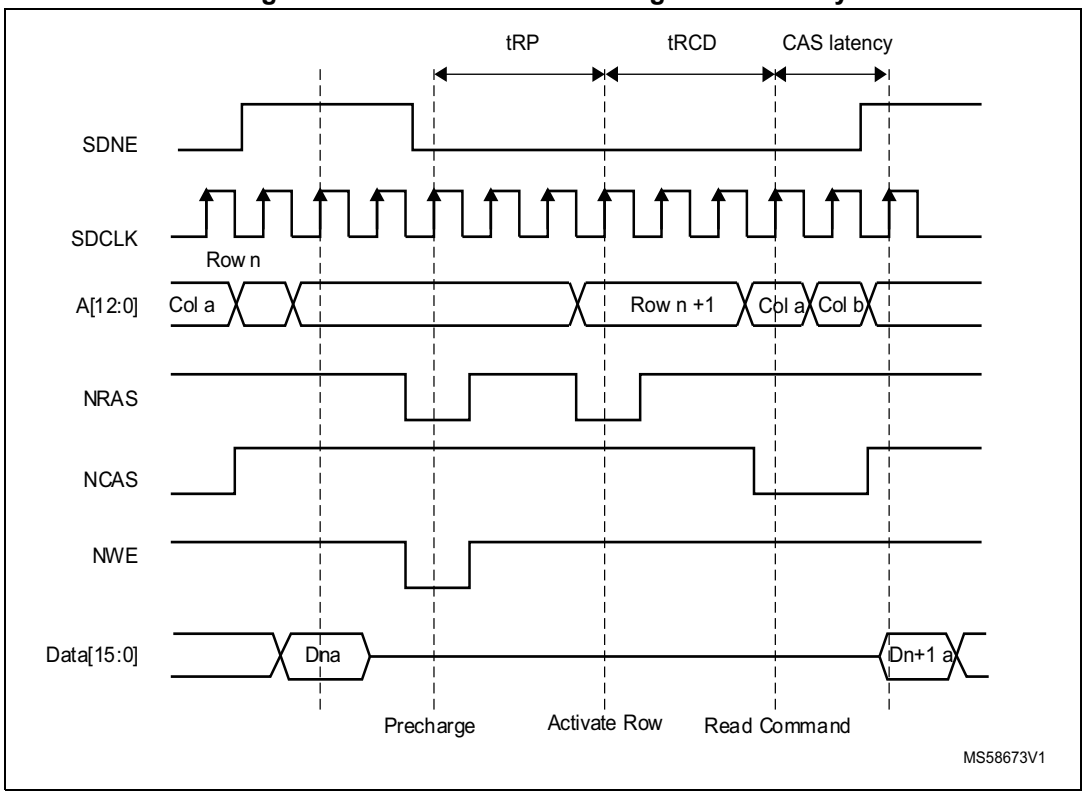
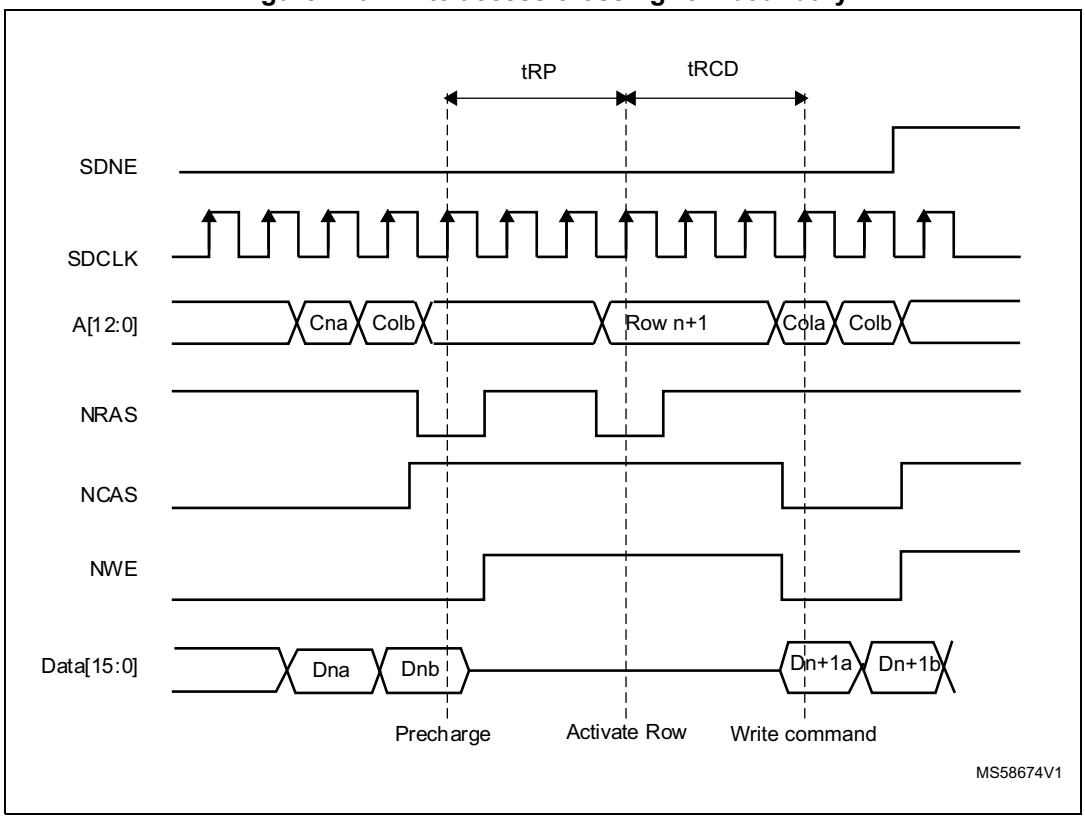


Figure 226. Write access crossing row boundary



If the next access is sequential and the current access crosses a bank boundary, the SDRAM controller activates the first row in the next bank and initiates a new read/write command. Two cases are possible:

- If the current bank is not the last one, the active row in the new bank must be precharged.At a bank boundary, the automatic activation of the next row is supported for all rows/columns and data bus width configuration.
- If the current bank is the last one and the selected SDRAM device is connected to Bank 1, the automatic activation of the next row in device connected to SDRAM Bank2 is not supported. A PALL software command must be issused on Bank1 before any any access on Bank2.

SDRAM controller refresh cycle

The Auto-refresh command is used to refresh the SDRAM device content. The SDRAM controller periodically issues auto-refresh commands. An internal counter is loaded with the COUNT value in the register FMC_SDRTR. This value defines the number of memory clock cycles between the refresh cycles (refresh rate). When this counter reaches zero, an internal pulse is generated.

If a memory access is ongoing, the auto-refresh request is delayed. However, if the memory access and the auto-refresh requests are generated simultaneously, the auto-refresh request takes precedence.

If the memory access occurs during an auto-refresh operation, the request is buffered and processed when the auto-refresh is complete.

If a new auto-refresh request occurs while the previous one was not served, the RE (Refresh Error) bit is set in the Status register. An Interrupt is generated if it has been enabled (REIE = ‘1’).

If SDRAM lines are not in idle state (not all row are closed), the SDRAM controller generates a PALL (Precharge ALL) command before the auto-refresh.

If the Auto-refresh command is generated by the FMC_SDCMR Command Mode register (Mode bits = ‘011’), a PALL command (Mode bits =’ 010’) must be issued first.

21.9.4 Low-power modes

Two low-power modes are available:

- Self-refresh mode

  The auto-refresh cycles are performed by the SDRAM device itself to retain data without external clocking.
- Power-down mode

  The auto-refresh cycles are performed by the SDRAM controller.

Self-refresh mode

This mode is selected by setting the MODE bits to ‘101’ and by configuring the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register.

The SDRAM clock stops running after a TRAS delay and the internal refresh timer stops counting only if one of the following conditions is met:

- A Self-refresh command is issued to both devices
- One of the devices is not activated (SDRAM bank is not initialized).

Before entering Self-Refresh mode, the SDRAM controller automatically issues a PALL command.

If the Write data FIFO is not empty, all data are sent to the memory before activating the Self-refresh mode and the BUSY status flag remains set.

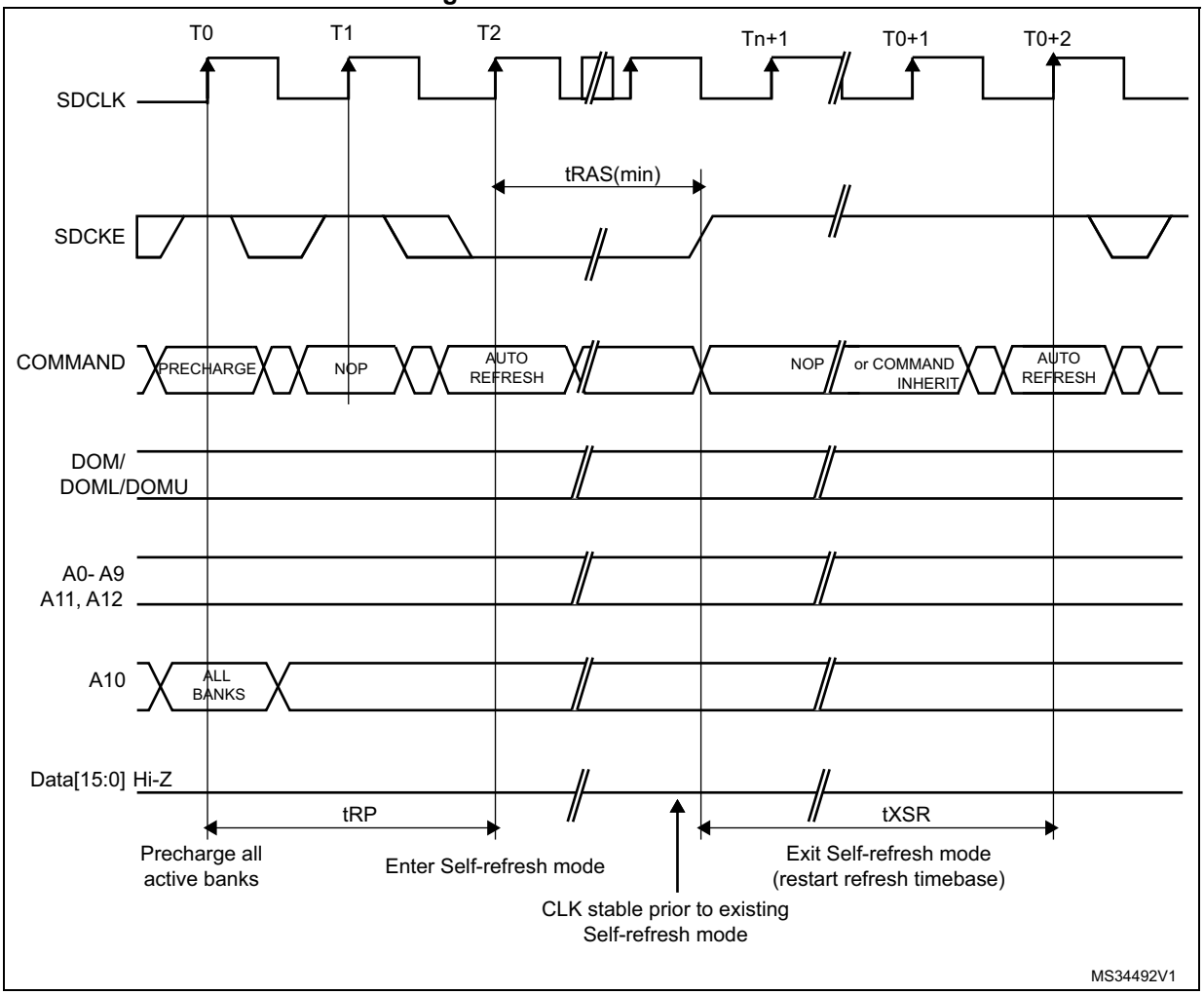
In Self-refresh mode, all SDRAM device inputs become don't care except for SDCKE which remains low.

The SDRAM device must remain in Self-refresh mode for a minimum period of time of TRAS and can remain in Self-refresh mode for an indefinite period beyond that. To guarantee this minimum period, the BUSY status flag remains high after the Self-refresh activation during a TRAS delay.

As soon as an SDRAM device is selected, the SDRAM controller generates a sequence of commands to exit from Self-refresh mode. After the memory access, the selected device remains in Normal mode.

To exit from Self-refresh, the MODE bits must be set to '000' (Normal mode) and the Target Bank bits (CTB1 and/or CTB2) must be configured in the FMC_SDCMR register.

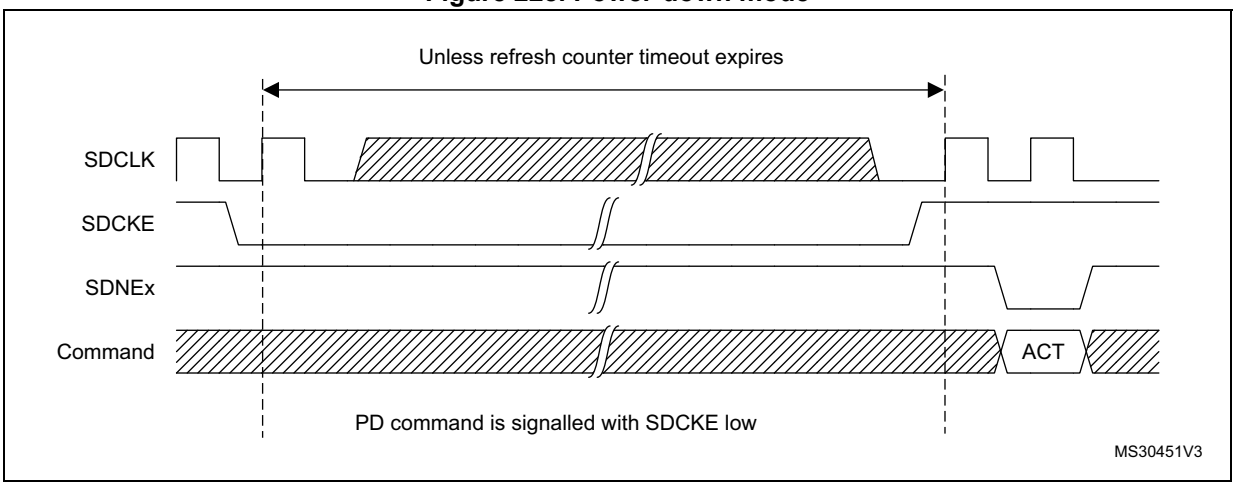
Figure 227. Self-refresh mode



Power-down mode

This mode is selected by setting the MODE bits to ‘110’ and by configuring the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register.

Figure 228. Power-down mode



If the Write data FIFO is not empty, all data are sent to the memory before activating the Power-down mode.

As soon as an SDRAM device is selected, the SDRAM controller exits from the Power-down mode. After the memory access, the selected SDRAM device remains in Normal mode.

During Power-down mode, all SDRAM device input and output buffers are deactivated except for the SDCKE which remains low.

The SDRAM device cannot remain in Power-down mode longer than the refresh period and cannot perform the Auto-refresh cycles by itself. Therefore, the SDRAM controller carries out the refresh operation by executing the operations below:

1. Exit from Power-down mode and drive the SDCKE high
2. Generate the PALL command only if a row was active during Power-down mode
3. Generate the auto-refresh command
4. Drive SDCKE low again to return to Power-down mode.

To exit from Power-down mode, the MODE bits must be set to ‘000’ (Normal mode) and the Target Bank bits (CTB1 and/or CTB2) must be configured in the FMC_SDCMR register.

21.9.5 SDRAM controller registers

SDRAM Control registers for SDRAM memory bank x (FMC_SDCRx)

Address offset: 0x140+ 0x4* (x - 1), (x = 1 to 2)

Reset value: 0x0000 02D0

This register contains the control parameters for each SDRAM memory bank

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RPIPE[1:0]		RBURST	SDCLK[1:0]		WP	CAS[1:0]		NB	MWID[1:0]		NR[1:0]		NC[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:13 **RPIPE[1:0]:** Read pipe

These bits define the delay, in fmc_ker_ck clock cycles, for reading data after CAS latency.

00: No fmc_ker_ck clock cycle delay

01: One fmc_ker_ck clock cycle delay

10: Two fmc_ker_ck clock cycle delay

11: reserved.

Note: The corresponding bits in the FMC_SDCR2 register is read only.

Bit 12 **RBURST:** Burst read

This bit enables Burst read mode. The SDRAM controller anticipates the next read commands during the CAS latency and stores data in the Read FIFO.

0: single read requests are not managed as bursts

1: single read requests are always managed as bursts

Note: The corresponding bit in the FMC_SDCR2 register is read only.

Bits 11:10 **SDCLK[1:0]:** SDRAM clock configuration

These bits define the SDRAM clock period for both SDRAM banks and allow disabling the clock before changing the frequency. In this case the SDRAM must be re-initialized.

00: SDCLK clock disabled

01: Reserved

10: SDCLK period = 2 x fmc_ker_ck periods

11: SDCLK period = 3 x fmc_ker_ck periods

Note: The corresponding bits in the FMC_SDCR2 register is read only.

Bit 9 **WP:** Write protection

This bit enables Write mode access to the SDRAM bank.

0: Write accesses allowed

1: Write accesses ignored

Bits 8:7 **CAS[1:0]:** CAS Latency

This bits sets the SDRAM CAS latency in number of memory clock cycles

00: reserved.

01: 1 cycle

10: 2 cycles

11: 3 cycles

Bit 6 **NB:** Number of internal banks

This bit sets the number of internal banks.

0: Two internal Banks

1: Four internal Banks

Bits 5:4 **MWID[1:0]:** Memory data bus width.

These bits define the memory device width.

00: 8 bits

01: 16 bits

10: 32 bits

11: reserved.

Bits 3:2 **NR[1:0]:** Number of row address bits
These bits define the number of bits of a row address.
00: 11 bit
01: 12 bits
10: 13 bits
11: reserved.

Bits 1:0 **NC[1:0]:** Number of column address bits
These bits define the number of bits of a column address.
00: 8 bits
01: 9 bits
10: 10 bits
11: 11 bits.

Note: Before modifying the RBURST or RPIPE settings or disabling the SDCLK clock, the user must first send a PALL command to make sure ongoing operations are complete.

SDRAM Timing registers for SDRAM memory bank x (FMC_SDTRx)

Address offset: 0x148 + 0x4 * (x - 1), (x = 1 to 2)

Reset value: 0x0FFF FFFF

This register contains the timing parameters of each SDRAM bank

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

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TRCD[3:0]:** Row to column delay
These bits define the delay, tRCD, from the Activate command to a Read/Write command in number of memory clock cycles.
0000: 1 cycles
0001: 2 cycles
....
1111: 16 cycles
Note: tRCD delay is replaced by (TWR[3:0]+1) cycles when TWR[3:0] > TRCD[3:0].

Bits 23:20 **TRP[3:0]:** Row precharge delay
These bits define the delay, tRP, from a Precharge command to another command in number of memory clock cycles. The TRP timing is only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRP must be programmed with the timing of the slowest device.
0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles
Note: The corresponding bits in the FMC_SDTR2 register are don't care.

Bits 19:16 **TWR[3:0]:** Recovery delay

These bits define the delay, tWR, between a Write and a Precharge command in number of memory clock cycles.

0000: 1 cycle

0001: 2 cycles

....

1111: 16 cycles

Note: TWR is a minimum of 2 cycles upon an auto refresh request

Note: TWR must be programmed to match the write recovery time (t_{WR}) defined in the SDRAM datasheet, and to guarantee that:

TWR ≥ TRAS - TRCD and TWR ≥ TRC - TRCD - TRP

Example: TRAS= 4 cycles, TRCD= 2 cycles. So, TWR ≥ 2 cycles. TWR must be programmed to 0x1.

If two SDRAM devices are used, the FMC_SDTR1 and FMC_SDTR2 must be programmed with the same TWR timing corresponding to the slowest SDRAM device.

If only one SDRAM device is used, the TWR timing must be kept at reset value (0xF) for the not used bank.

Bits 15:12 **TRC[3:0]:** Row cycle delay

These bits define the delay, tRC, from the Refresh command to the Activate command, as well as the delay between two consecutive Refresh commands. It is expressed in number of memory clock cycles. The TRC timing is only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRC must be programmed with the timings of the slowest device.

0000: 1 cycle

0001: 2 cycles

....

1111: 16 cycles

Note: TRC must match the TRC and TRFC (Auto Refresh period) timings defined in the SDRAM device datasheet.

Note: The corresponding bits in the FMC_SDTR2 register are don't care.

Bits 11:8 **TRAS[3:0]:** Self refresh time

These bits define the minimum Self-refresh period in number of memory clock cycles.

0000: 2 cycle

0001: 3 cycles

....

1111: 17 cycles

Bits 7:4 **TXSR[3:0]:** Exit Self-refresh delay

These bits define the delay, tXSR, from releasing the Self-refresh command to issuing the Activate command in number of memory clock cycles.

0000: 1 cycle

0001: 2 cycles

....

1111: 16 cycles

Note: tXSR delay is replaced by (TWR[3:0]+1) cycles when TWR[3:0] > TXSR[3:0].

Note: If two SDRAM devices are used, the FMC_SDTR1 and FMC_SDTR2 must be programmed with the same TXSR timing corresponding to the slowest SDRAM device.

Bits 3:0 **TMRD[3:0]:** Load Mode Register to Active

These bits define the delay between a Load Mode Register command and an Active or Refresh command in number of memory clock cycles.
0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles

Note: If two SDRAM devices are connected, all the accesses performed simultaneously to both devices by the Command Mode register (Load Mode Register command) are issued using the timing parameters configured for Bank 1 (TMRD and TRAS timings) in the FMC_SDTR1 register.

The TRP and TRC timings are only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRP and TRC timings must be programmed with the timings of the slowest device.

SDRAM Command mode register (FMC_SDCMR)

Address offset: 0x150

Reset value: 0x0000 0000

This register contains the command issued when the SDRAM device is accessed. This register is used to initialize the SDRAM device, and to activate the Self-refresh and the Power-down modes. As soon as the MODE field is written, the command is issued only to one or to both SDRAM banks according to CTB1 and CTB2 command bits. This register is the same for both SDRAM banks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MRD[13:7]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MRD[6:0]							NRFS[3:0]				CTB1	CTB2	MODE[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	w	w	w	w

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

Bits 22:9 **MRD[13:0]:** Mode Register definition

This 14-bit field defines the SDRAM Mode Register content. The Mode Register is programmed using the Load Mode Register command. The MRD[13:0] bits are also used to program the extended mode register for mobile SDRAM.

Bits 8:5 **NRFS[3:0]**: Number of Auto-refresh

These bits define the number of consecutive Auto-refresh commands issued when MODE = ‘011’.
0000: 1 Auto-refresh cycle
0001: 2 Auto-refresh cycles
....
1110: 15 Auto-refresh cycles
1111: 16 Auto-refresh cycles

Bit 4 **CTB1:** Command Target Bank 1

This bit indicates whether the command is issued to SDRAM Bank 1 or not.
0: Command not issued to SDRAM Bank 1
1: Command issued to SDRAM Bank 1

Bit 3 **CTB2:** Command Target Bank 2

This bit indicates whether the command is issued to SDRAM Bank 2 or not.

0: Command not issued to SDRAM Bank 2

1: Command issued to SDRAM Bank 2

Bits 2:0 **MODE[2:0]:** Command mode

These bits define the command issued to the SDRAM device.

000: Normal Mode

001: Clock Configuration Enable

010: PALL (“All Bank Precharge”) command

011: Auto-refresh command

100: Load Mode Register

101: Self-refresh command

110: Power-down command

111: Reserved

Note: When a command is issued, at least one Command Target Bank bit (CTB1 or CTB2) must be set otherwise the command is ignored.

Note: If two SDRAM banks are used, the Auto-refresh and PALL command must be issued simultaneously to the two devices with CTB1 and CTB2 bits set otherwise the command is ignored.

Note: If only one SDRAM bank is used and a command is issued with it’s associated CTB bit set, the other CTB bit of the unused bank must be kept to 0.

SDRAM refresh timer register (FMC_SDRTR)

Address offset: 0x154

Reset value: 0x0000 0000

This register sets the refresh rate in number of SDCLK clock cycles between the refresh cycles by configuring the Refresh Timer Count value.

$$\text{Refresh rate} = (\text{COUNT} + 1) \times \text{SDRAM clock frequency}$$

$$\text{COUNT} = (\text{SDRAM refresh period} / \text{Number of rows}) - 20$$

Below an example of refresh rate calculation:

$$\text{Refresh rate} = 64\ \text{ms} / (8196\text{rows}) = 7.81\mu\text{s}$$

where 64 ms is the SDRAM refresh period.

$$7.81\mu\text{s} \times 60\text{MHz} = 468.6$$

The refresh rate must be increased by 20 SDRAM clock cycles (as in the above example) to obtain a safe margin if an internal refresh request occurs when a read request has been accepted. It corresponds to a COUNT value of ‘0000111000000’ (448).

This 13-bit field is loaded into a timer which is decremented using the SDRAM clock. This timer generates a refresh pulse when zero is reached. The COUNT value must be set at least to 41 SDRAM clock cycles.

As soon as the FMC_SDRTR register is programmed, the timer starts counting. If the value programmed in the register is ’0’, no refresh is carried out. This register must not be reprogrammed after the initialization procedure to avoid modifying the refresh rate.

Each time a refresh pulse is generated, this 13-bit COUNT field is reloaded into the counter.

If a memory access is in progress, the Auto-refresh request is delayed. However, if the memory access and Auto-refresh requests are generated simultaneously, the Auto-refresh takes precedence. If the memory access occurs during a refresh operation, the request is buffered to be processed when the refresh is complete.

This register is common to SDRAM bank 1 and bank 2.

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

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **REIE:** RES Interrupt Enable

0: Interrupt is disabled

1: An Interrupt is generated if RE = 1

Bits 13:1 **COUNT[12:0]:** Refresh Timer Count

This 13-bit field defines the refresh rate of the SDRAM device. It is expressed in number of memory clock cycles. It must be set at least to 58 SDRAM clock cycles (0x34).

Refresh rate = (COUNT + 1) x SDRAM frequency clock

COUNT = (SDRAM refresh period / Number of rows) - 20

Bit 0 **CRE:** Clear Refresh error flag

This bit is used to clear the Refresh Error Flag (RE) in the Status Register.

0: no effect

1: Refresh Error flag is cleared

Note: The programmed COUNT value must not be equal to the sum of the following timings: TWR+TRP+TRC+TRCD+4 memory clock cycles .

SDRAM Status register (FMC_SDSR)

Address offset: 0x158

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODES2[1:0]		MODES1[1:0]		RE
											r	r	r	r	r

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

Bits 4:3 **MODES2[1:0]:** Status Mode for Bank 2
These bits define the Status Mode of SDRAM Bank 2.
00: Normal Mode
01: Self-refresh mode
10: Power-down mode

Bits 2:1 **MODES1[1:0]:** Status Mode for Bank 1
These bits define the Status Mode of SDRAM Bank 1.
00: Normal Mode
01: Self-refresh mode
10: Power-down mode

Bit 0 **RE:** Refresh error flag
0: No refresh error has been detected
1: A refresh error has been detected
An interrupt is generated if REIE = 1 and RE = 1

21.9.6 FMC register map

The following table summarizes the FMC registers.

Table 198. FMC register map

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**FMC_BCR1**	FMCEN	Res.	Res.	Res.	Res.	Res.	BMAP[1:0]		Res.	Res.	WFDIS	CCLKEN	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID		MTYP		MUXEN	MBKEN
0x08	**FMC_BCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBURSTRW	CPSIZE [2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID [1:0]		MTYP [1:0]		MUXEN	MBKEN
	Reset value													0	0	0	0	0	0	1	1	0		0	0		1	0	1	0	0	1	0

Table 198. FMC register map (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	**FMC_BCR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBURSTRW	CPSIZE [2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID [1:0]		MTYP [1:0]		MUXEN	MBKEN
	Reset value													0	0	0	0	0	0	1	1	0		0	0		1	0	1	0	0	1	0
0x18	**FMC_BCR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBURSTRW	CPSIZE [2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID [1:0]		MTYP [1:0]		MUXEN	MBKEN
	Reset value													0	0	0	0	0	0	1	1	0		0	0		1	0	1	0	0	1	0
0x04	**FMC_BTR1**	Res.	Res.	ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value			0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0C	**FMC_BTR2**	Res.	Res.	ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value			0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	**FMC_BTR3**	Res.	Res.	ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value			0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x1C	**FMC_BTR4**	Res.	Res.	ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value			0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x104	**FMC_BWTR1**	Res.	Res.	ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value			0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x10C	**FMC_BWTR2**	Res.	Res.	ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value			0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x114	**FMC_BWTR3**	Res.	Res.	ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value			0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x11C	**FMC_BWTR4**	Res.	Res.	ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value			0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x80	**FMC_PCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCPS [2:0]				TAR[3:0]				TCLR[3:0]				Res.	Res.	ECCEN	PWID [1:0]		Res.	PBKEN	PWAITEN	Res.
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0			0	0	1	1	0	0	
0x84	**FMC_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS
	Reset value																										1	0	0	0	0	0	0
0x88	**FMC_PMEM**	MEMHIZx[7:0]								MEMHOLDx[7:0]								MEMWAITx[7:0]								MEMSETx[7:0]							
	Reset value	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0
0x8C	**FMC_PATT**	ATTHIZ[7:0]								ATTHOLD[7:0]								ATTWAIT[7:0]								ATTSET[7:0]							
	Reset value	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0
0x94	**FMC_ECCR**	ECC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x140	**FMC_SDCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIPE[1:0]		RBURST	SDCLK [1:0]		WP	CAS [1:0]		NB	MWID [1:0]		NR[1:0]		NC	
	Reset value																		0	0	0	1	1	0	1	0	0	1	0	0	0	0	0

Table 198. FMC register map (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x144	**FMC_SDCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIPE [1:0]		RBURST	SDCLK [1:0]		WP	CAS [1:0]		NB	MWID [1:0]		NR[1:0]		NC	
	Reset value																		0	0	0	1	1	0	1	0	0	1	0	0	0	0	0
0x148	**FMC_SDTR1**	Res.	Res.	Res.	Res.	TRCD[3:0]				TRP[3:0]				TWR[3:0]				TRC[3:0]				TRAS[3:0]				TXSR[3:0]				TMRD[3:0]			
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14C	**FMC_SDTR2**	Res.	Res.	Res.	Res.	TRCD[3:0]				TRP[3:0]				TWR[3:0]				TRC[3:0]				TRAS[3:0]				TXSR[3:0]				TMRD[3:0]			
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x150	**FMC_SDCMR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MRD														NRFS[3:0]				CTB1	CTB2	MODE[2:0]		
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x154	**FMC_SDRTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REIE	COUNT[12:0]													CRE	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x158	**FMC_SDSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODES2[1:0]		MODES1[1:0]		RE
	Reset value																												0	0	0	0	0

Refer to *Section 2.3 on page 132* for the register boundary addresses.

22 Extended-SPI interface (XSPI)

22.1 XSPI introduction

The XSPI supports most external serial memories such as serial PSRAMs, serial NAND and serial NOR flash memories, HyperRAM™ and HyperFlash™ memories, with the following functional modes:

- indirect mode: all the operations are performed using the XSPI registers to preset commands, addresses, data, and transfer parameters.
- automatic status-polling mode: the external memory status register is periodically read and an interrupt can be generated in case of flag setting. This feature is only available in regular-command protocol.
- memory-mapped mode: the external memory is memory mapped and it is seen by the system as if it was an internal memory, supporting both read and write operations.

The XSPI supports the following protocol with associated frame format:

- the regular-command frame format with the command, address, alternate byte, dummy cycles, and data phase
- the HyperBus™ frame format

22.2 XSPI main features

- Functional modes: indirect, automatic status-polling, and memory-mapped
- Read and write support in memory-mapped mode
- External (P)SRAM memory support
- Support for single, dual, quad, octal communication
- Dual-memory configuration, where eight bits can be sent/received simultaneously by accessing two quad memories in parallel
- SDR (single-data rate) and DTR (double-transfer rate) support
- Data strobe support
- Fully programmable opcode
- Fully programmable frame format
- Support wrapped-type access to memory in read direction
- HyperBus support
- Integrated FIFO for reception and transmission
- Asynchronous bus clock versus kernel clock support
- 8-, 16-, and 32-bit data accesses allowed
- DMA protocol support
- DMA channel for indirect mode operations
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error
- AXI acceptance 2: acceptance is offered based on a read transaction followed by a second write or read request (acknowledged on the bus), while the first one is being processed.
- Dual chip select support (NCS1 and NCS2)

- Extended external memory support: if two same size external memories (extmem1 and extmem2) are connected to the same I/O port, contiguously in the memory map and driven by a single XSPI. This XSPI automatically switches CS to extmem1 or extmem2, according to the address on interconnect side.
- Possibility to disable the automatic prefetch

22.3 XSPI implementation

Table 199. XSPI implementation

Feature	XSPI1
Octal mode configuration (8-bit)	X
16-bit mode configuration	-
HyperBus standard compliant	X
Xccela standard compliant	X
XSPI (JESD251C) standard compliant	X
AMBA® AXI compliant data interface	X
Asynchronous AXI clock versus kernel clock	X
Functional modes: indirect, automatic status-polling, and memory-mapped	X
Dual chip select support (NCS1 and NCS2)	X
Read and write support in memory-mapped mode	X
Dual-quad configuration	X
Dual-octal configuration	-
SDR (single-data rate) and DTR (double-transfer rate)	X
Data strobe (DS, DQS)	X
Fully programmable opcode	X
Fully programmable frame format	X
Integrated FIFO for reception and transmission	X
8, 16, and 32-bit data accesses	X
Interrupt on FIFO threshold, timeout, operation complete, and access error	X
Bus error bit	X
Address offset	-
Compliant with dual-XSPI arbiter (communication regulation)	-
Extended CSHT timeout	X
Memory-mapped write	X
Refresh counter	X
High-speed interface	-
GP/HPDMA interface	X
AXI acceptance 2	X

Table 199. XSPI implementation (continued)

Feature	XSPI1
Extended external memory support	X
Automatic prefetch that can be disabled	X
Prefetch disable	X
Prefetch hardware software	X

22.4 XSPI functional description

22.4.1 XSPI block diagram

Figure 229. XSPI block diagram for octal configuration

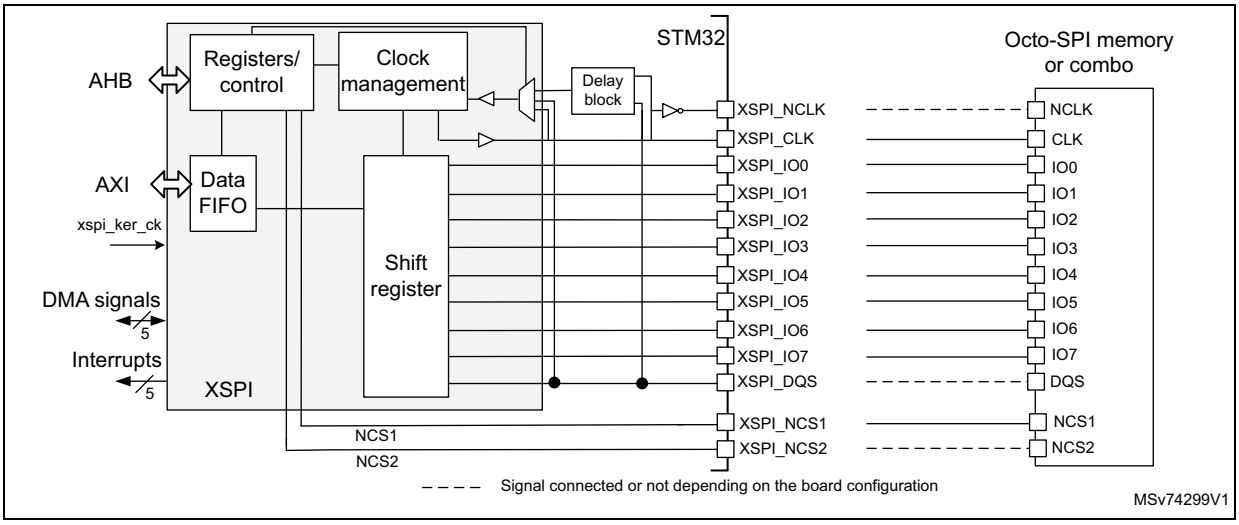


Figure 230. XSPI block diagram for quad configuration

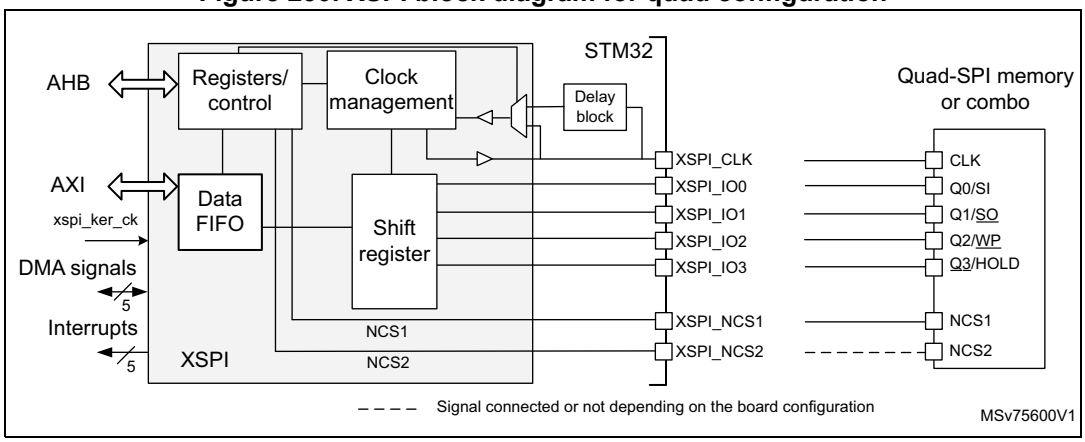
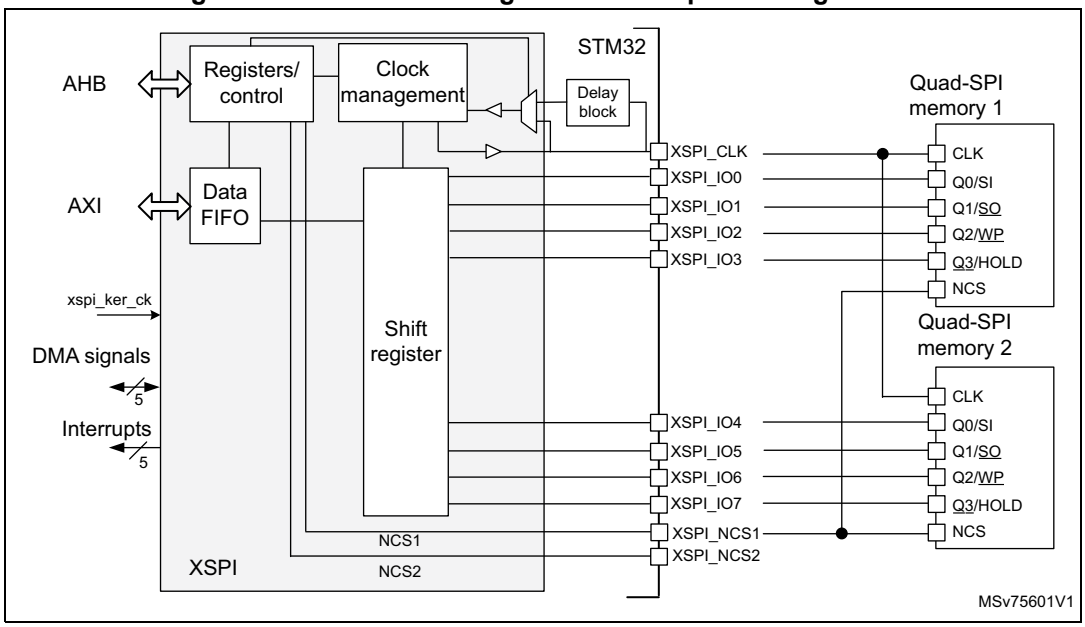


Figure 231. XSPI block diagram for dual-quad configuration



22.4.2 XSPI pins and internal signals

Table 200. XSPI input/output pins

Pin name	Type	Description
XSPI_NCLK	Output	XSPI inverted clock to support 1.8 V HyperBus protocol
XSPI_CLK	Output	XSPI clock
XSPI_IOn (n = 0 to 7)	Input/output	XSPI data pins
XSPI_NCS1,2	Output	Chip select for the memory
XSPI_DQS	Input/output	Data strobe/write mask signal from/to the memory

Caution: Use the same configuration (output speed, HSLV[(a)]) for all XSPI input/output pins to avoid any data corruption.

Table 201. XSPI internal signals

Signal name	Type	Description
xspi_hclk	Input	XSPI AHB clock
xspi_aclk	Input	XSPI AXI clock
xspi_ker_ck	Input	XSPI kernel clock
xspi_dma	N/A	DMA request signal
xspi_it	Output	Global interrupt line (see *Table 205* for the multiple sources of interrupt)

a. When applicable.

22.4.3 Clock constraints

The XSPI AXI clock (xspi_aclk) has to be greater or equal than the XSPI kernel clock (xspi_ker_ck).

22.4.4 XSPI interface to memory modes

The XSPI supports the following protocol:

- regular-command protocol
- HyperBus protocol

The XSPI uses from 6 to 12 signals to interface with a memory, depending on the functional mode:

- NCS: chip-select (in the following sections, unless otherwise specified, NCS applies for NCS1 or NCS2. Only one of these two signals is active at a given moment in time)
- CLK: communication clock
- NCLK: inverted clock used only in the 1.8 V HyperBus protocol
- DQS: data strobe used only in regular-command protocol as input only
- IO[3:0]: data bus LSB
- IO[7:4]:
  - data bus MSB used in dual-quad and octal configurations
  - data bus can be used as possible remap for quad-SPI mode

22.4.5 XSPI regular-command protocol

When in regular-command protocol, the XSPI communicates with the external device using commands. Each command can include the following phases:

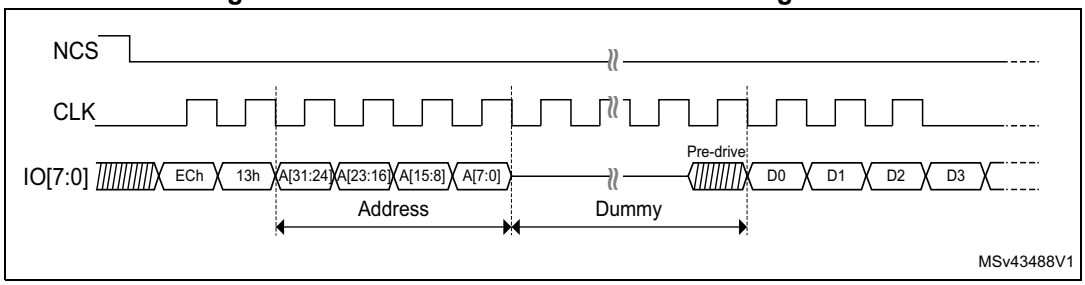
- Instruction phase
- Address phase
- Alternate-byte phase
- Dummy-cycle phase
- Data phase

Any of these phases can be configured to be skipped but, in case of single-phase command, the only use case supported is instruction-phase-only.

The NCS falls before the start of each command and rises again after each command finishes.

In memory-mapped mode, both read and write operations are supported: as a consequence, some of the configuration registers are duplicated to specify write operations (read operations are configured using regular registers).

Figure 232. SDR read command in octal configuration



The specific regular-command protocol features are configured through the registers in the 0x0100-0x01FC offset range.

Instruction phase

During this phase, a 1- to 4-byte instruction is sent to the external device specifying the type of operation to be performed. The size of the instruction to be sent is configured by ISIZE[1:0] in XSPI_CCR and the instruction is programmed in INSTRUCTION[31:0] of XSPI_IR.

The instruction phase can optionally send:

- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using IMODE[2:0] of XSPI_CCR.

The instruction can be sent in DTR mode on each rising and falling edge of the clock, by setting IDTR in XSPI_CCR.

When IMODE[2:0] = 000 in XSPI_CCR, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

In memory-mapped mode, the instruction used for the write operation is specified in XSPI_WIR, and the instruction format is specified in XSPI_WCCR. The instruction used for the read operation and the instruction format are specified in XSPI_IR and XSPI_CCR.

Address phase

In the address phase, 1 to 4 bytes are sent to the external device, to indicate the address of the operation. The number of address bytes to be sent is configured by ADSIZE[1:0] in XSPI_CCR.

In indirect and automatic status-polling modes, the address bytes to be sent are specified by ADDRESS[31:0] in XSPI_AR. In memory-mapped mode, the address is given directly via the AXI (from any master in the system).

The address phase can send:

- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using ADMODE[2:0] in XSPI_CCR.

The address can be sent in DTR mode (on each rising and falling edge of the clock) setting ADDTR in XSPI_CCR.

When ADMODE[2:0] = 000, the address phase is skipped and the command sequence proceeds directly to the next phase, if any.

In memory-mapped mode, the address format for the write operation is specified in XSPI_WCCR. The address format for the read operation is specified in XSPI_CCR.

Alternate-byte phase

In the alternate-byte phase, 1 to 4 bytes are sent to the external device, generally to control the mode of operation. The number of alternate bytes to be sent is configured by ABSIZE[1:0] in XSPI_CCR. The bytes to be sent are specified in XSPI_ABR.

The alternate-byte phase can send:

- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using ABMODE[2:0] in XSPI_CCR.

The alternate bytes can be sent in DTR mode (on each rising and falling edge of the clock) setting ABDTR in XSPI_CCR.

When ABMODE[2:0] = 000, the alternate-byte phase is skipped and the command sequence proceeds directly to the next phase, if any.

There may be times when only a single nibble needs to be sent during the alternate-byte phase rather than a full byte, such as when the dual-SPI mode is used and only two cycles are used for the alternate bytes.

In this case, the firmware can use the quad-SPI mode (ABMODE[2:0] = 011) and send a byte with bits 7 and 3 of ALTERNATE[31:0] set to 1 (keeping the IO3 line high), and bits 6 and 2 set to 0 (keeping the IO2 line low), in XSPI_IR.

The upper two bits of the nibble to be sent are then placed in bits 5:4 of ALTERNATE[31:0] while the lower two bits are placed in bits 1:0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE[31:0] must be set to 0x8A (1000_1010).

In memory-mapped mode, the alternate bytes used for the write operation are specified in XSPI_WABR, and the alternate byte format is specified in XSPI_WCCR. The alternate bytes used for read operation and the alternate byte format are specified in XSPI_ABR and XSPI_CCR.

Dummy-cycle phase (memory latency)

In the dummy-cycle phase, 1 to 31 cycles are given without any data being sent or received, in order to give the external device, the time to prepare for the data phase when higher clock frequencies are used. The number of cycles given during this phase is specified by DCYC[4:0] in XSPI_TCR. In both SDR and DTR modes, the duration is specified as a number of full CLK cycles.

When DCYC[4:0] = 00000, the dummy-cycle phase is skipped, and the command sequence proceeds directly to the data phase, if present.

In order to assure enough “turn-around” time for changing the data signals from the output mode to the input mode, there must be at least one dummy cycle when using the dual-, the quad-, or the octal-SPI mode, to receive data from the external device.

In memory-mapped mode, the dummy cycles for the write operations are specified in XSPI_WTCR. The dummy cycles for the read operation are specified in XSPI_TCR.

Data phase

During the data phase, any number of bytes can be sent to or received from the external device.

In indirect mode, the number of bytes to be sent/received is specified in XSPI_DLR. In this mode, the data to be sent to the external device must be written to XSPI_DR, while in indirect-read mode the data received from the external device is obtained by reading XSPI_DR.

In automatic status-polling mode, the number of bytes to be received is specified in XSPI_DLR, and the data received from the external device can be obtained by reading XSPI_DR.

In memory-mapped mode, the data read or written, is sent or received directly over the AXI to the Cortex core or to a DMA.

The data phase can send/receive:

- 1 bit at a time (over IO0/IO1 (SO/SI respectively) in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using DMODE[2:0] in XSPI_CCR.

The data can be sent or received in DTR mode (on each rising and falling edge of the clock) setting DDTR in XSPI_CCR.

When DMODE[2:0] = 000, the data phase is skipped, and the command sequence finishes immediately by raising the NCS. This configuration must be used only in indirect-write mode.

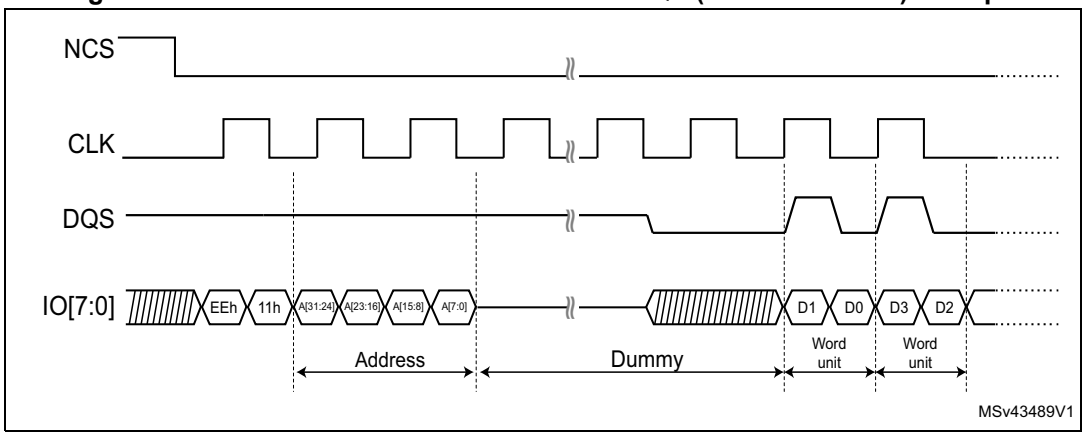
In memory-mapped mode, the data format for the write operation is specified in XSPI_WCCR. The data format for the read operation is specified in XSPI_CCR.

DQS use

The DQS signal can be used for data strobing during the read transactions when the device toggles the DQS aligned with the data.

The DQS management can be enabled by setting DQSE in XSPI_CCR.

Figure 233. DTR read in octal-SPI mode with DQS (Macronix mode) example



22.4.6 XSPI regular-command protocol signal interface

Single-SPI mode

The legacy SPI mode allows just a single bit to be sent/received serially. In this mode, the data is sent to the external device over the SO signal (Single-SPI Output) (whose I/Os are shared with IO0). The data received from the external device arrives via SI (Single-SPI Input) (whose I/Os are shared with IO1).

Compared to the SPI legacy mode, I0/SO and I1/SI are respectively equivalent to MOSI and MISO, having the XSPI generating the clock.

The different phases can each be configured separately to use this single-SPI mode by setting to 001 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in single-SPI mode:

- IO0 (SO) is in output mode.
- IO1 (SI) is in input mode (high impedance).
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1 (to deactivate the “hold” function).
- IO4 to IO7 are in output mode and forced to 0.

This is the case even for the dummy phase if DMODE[2:0] = 001.

Dual-SPI mode

In dual-SPI mode, two bits are sent/received simultaneously over the IO0/IO1 signals.

The different phases can each be configured separately to use dual-SPI mode by setting to 010 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in dual-SPI mode:

- IO0/IO1 are at high-impedance (input) during the data phase for the read operations, and outputs in all other cases.
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase, when DMODE[2:0] = 010, IO0 and IO1 are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

Quad-SPI mode

In quad-SPI mode, four bits are sent/received simultaneously over the IO0/IO1/IO2/IO3 signals.

The different phases can each be configured separately to use the quad-SPI mode by setting to 011 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in quad-SPI mode:

- IO0 to IO3 are all at high-impedance (inputs) during the data phase for the read operations, and outputs in all other cases.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase, when DMODE[2:0] = 011, IO0 to IO3 are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

Octal-SPI mode

In regular octal-SPI mode, the eight bits are sent/received simultaneously over the IO[0:7] signals.

The different phases can each be configured separately to use the octal-SPI mode by setting to 100 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase that is configured in octal-SPI mode, IO[0:7] are all at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase, when DMODE[2:0] = 100, IO[0:7] are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

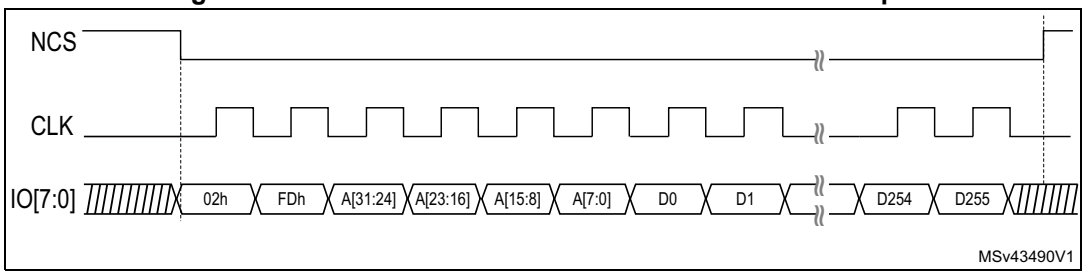
Single-data rate (SDR) mode

By default, all the phases operate in single-data rate (SDR) mode.

In SDR mode, when the XSPI drives the IO0/SO, IO1 to IO7 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the XSPI assumes that the external devices also send the data using CLK falling edge. By default (when SSHIFT = 0 in XSPI_TCR), the signals are sampled using the following (rising) edge of CLK.

Figure 234. SDR write command in octal-SPI mode example



Double-transfer rate (DTR) mode

Each of the instruction, address, alternate-byte, and data phases can be configured to operate in DTR mode setting IDTR, ADDTR, ABDTR, and DDTR in XSPI_CCR.

In memory-mapped mode, the DTR mode for each phase of the write operations is specified in XSPI_WCCR. The DTR mode for each phase of the read operations is specified in XSPI_CCR.

In DTR mode, when the XSPI drives the IO0/SO and IO1 to IO7 signals in the instruction, address, and alternate-byte phases, a bit is sent or received on each of the falling and rising edges of CLK.

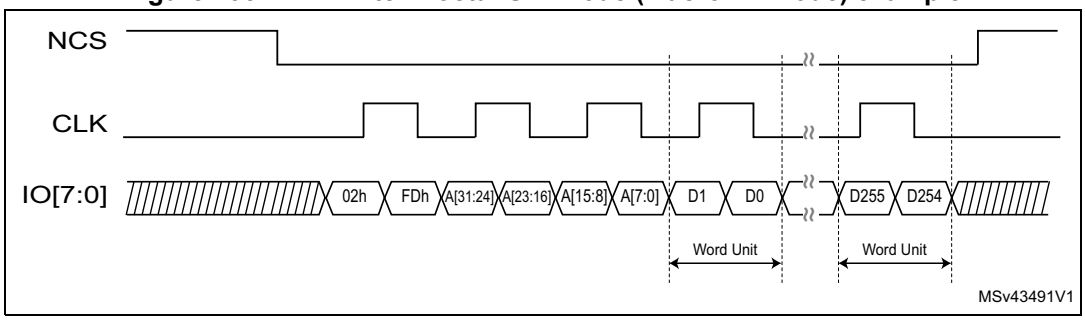
In DTR mode, when the XSPI drives the IO0 to IO7 signals in the data phases, a bit is sent or received on each of the falling and rising edges of CLK.

When receiving data in DTR mode, the XSPI assumes that the external devices also send the data using both CLK rising and falling edges. When DDTR = 1 in XSPI_CCR, the software must clear SSHIFT in XSPI_TCR. Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

In DTR mode, it is recommended to set DHQC of XSPI_TCR, to shift the outputs by a quarter of cycle and avoid holding issues on the memory side.

Note: DHQC must not be set when the prescaler value is 0, as this action leads to unpredictable behavior.

Figure 235. DTR write in octal-SPI mode (Macronix mode) example



Dual-quad configuration

When DMM = 1 in XSPI_CR, the XSPI is in dual-memory configuration: if DMODE = 011, two external quad-SPI devices (device A and device B) are used in order to send/receive eight bits (or 16 bits in DTR mode) every cycle, effectively doubling the throughput.

Each device (A or B) uses the same CLK and NCS signals, but each has separate IO0 to IO3 signals.

The dual-quad configuration can be used in conjunction with the single-SPI, dual-SPI, and quad-SPI modes, as well as with either the SDR or DTR mode.

The device size, as specified by DEVSIZE[4:0] in XSPI_DCR1, must reflect the total external device capacity that is the double of the size of one individual component.

If address X is even, then the byte that the XSPI gives for address X is the byte at the address X/2 of device A, and the byte that the XSPI gives for address X + 1 is the byte at the address X/2 of device B. In other words, the bytes at even addresses are all stored in device A and the bytes at odd addresses are all stored in device B.

When reading the status registers of the devices in dual-quad configuration, twice as many bytes must be read compared to the same read in regular-command protocol: if each device gives eight valid bits after the instruction for fetching the status register, then the XSPI must be configured with a data length of 2 bytes (16 bits), and the XSPI receives one byte from each device.

If each device gives a status of 16 bits, then the XSPI must be configured to read 4 bytes to get all the status bits of both devices in dual-quad configuration. The least-significant byte of the result (in the data register) is the least-significant byte of device A status register. The next byte is the least-significant byte of device B status register. Then, the third byte of the data register is the device A second byte. The fourth byte is the device B second byte (if devices have 16-bit status registers).

An even number of bytes must always be accessed in dual-quad configuration. For this reason, bit 0 of DL[31:0] in XSPI_DLR is stuck at 1 when DMM = 1.

In dual-quad configuration, the behavior of device A interface signals is basically the same as in normal mode. Device B interface signals have exactly the same waveforms as device A ones during the instruction, address, alternate-byte, and dummy-cycle phases. In other words, each device always receives the same instruction and the same address.

Then, during the data phase, the AIOx and the BIOx buses both transfer data in parallel, but the data that is sent to (or received from) device A is distinct than the one from device B.

22.4.7 HyperBus protocol

The XSPI can communicate with the external device using the HyperBus protocol.

The HyperBus uses 11 to 12 pins depending on the operating voltage:

- IO[7:0] as bidirectional data bus
- DQS for read and write data strobe and latency insertion
- NCS
- CLK
- NCLK for 1.8 V operations (to support this mode, the device must be powered with 1.8 V)

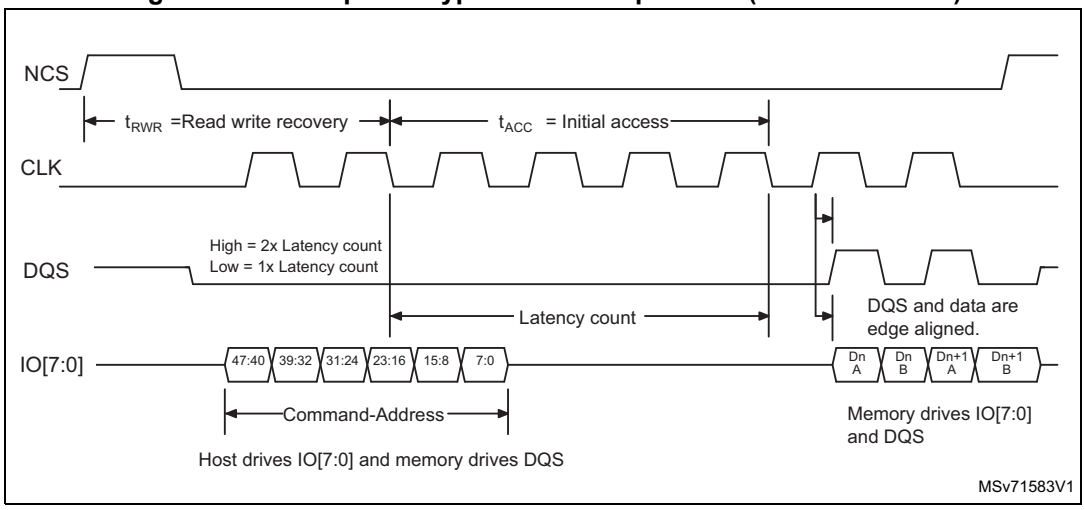
The HyperBus does not require any command specification nor any alternate bytes. As a consequence, a separate register set is used to define the timing of the transaction.

The HyperBus frame is composed of the following phases:

- Command/address phase
- Data phase

The NCS falls before the start of a transaction and rises again after each transaction finishes.

Figure 236. Example of HyperBus read operation (8-bit data mode)



The specific HyperBus features are configured through the registers in the 0x0200-0x02FC offset range.

Command/address phase

During this initial phase, the XSPI sends 48 bits over IO[7:0] to specify the operations to be performed with the external device.

Table 202. Command/address phase description

CA bit	Bit name	Description
47	R/W#	Identifies the transaction as a read or a write.
46	Address space	Indicates if the transaction accesses the memory or the register space.
45	Burst type	Indicates if the burst is linear or wrapped.
44-16	Row and upper column address	Selects the row and the upper column addresses.
15-3	Reserved	-
2-0	Lower column address	Selects the starting 16-bit word within the half page.

The address space is configured through the memory type MTYP[2:0] in XSPI_DCR1.

The total size of the device must be configured through DEVSIZE[4:0] of XSPI_DCR1. In case of multi-chip product (MCP), the device size is the sum of all the sizes of all the MCP dies.

Read/write operation with initial latency

The HyperBus read and write operations need to respect two timings:

- t_{RWR}: minimal read/write recovery time for the device (defined by TRWR[7:0] in XSPI_HLCR)
- t_{ACC}: access time for the device (defined by TACC[7:0] in XSPI_HLCR) according to the memory latency

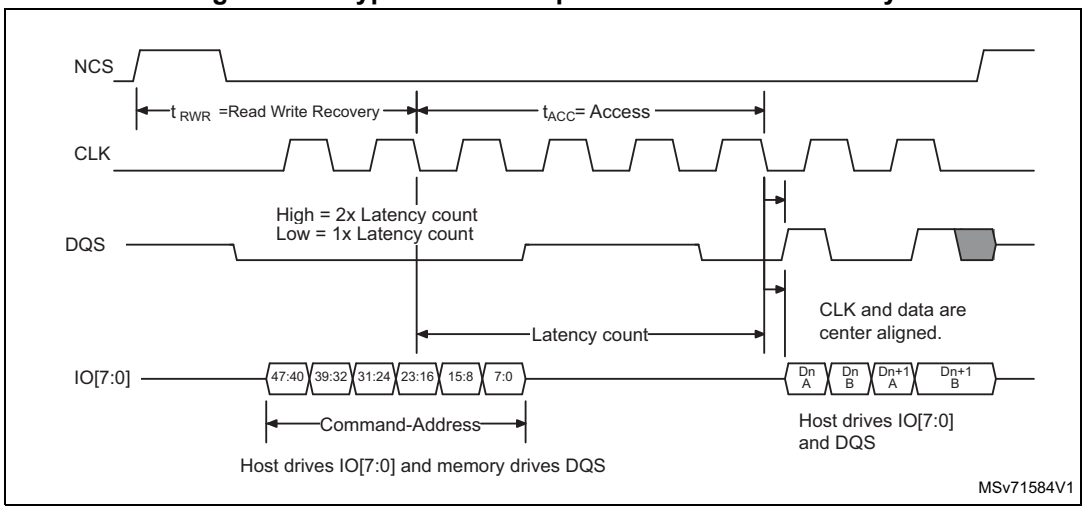
During the read operation, the DQS is used by the device, in two ways (see *Figure 236*):

- during the command/address phase, to request an additional latency
- during the data phase, for data strobing

During the write operation, the DQS is used:

- by the device, during the command/address phase, to request an additional latency.
- by the XSPI, during the data phase, for write data masking.

Figure 237. HyperBus write operation with initial latency



Read/write operation with additional latency

If the device needs an additional latency (during refresh period of an SDRAM for example), DQS must be tied to one during one of the DQS signals, during the command/address phase.

An additional t_{ACC} duration is added by the XSPI to meet the device request.

Figure 238. HyperBus read operation with additional latency

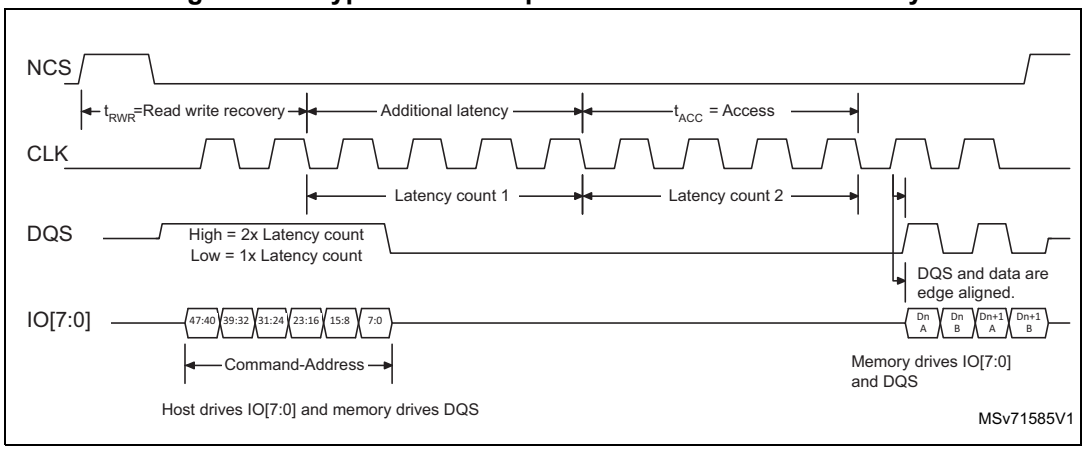
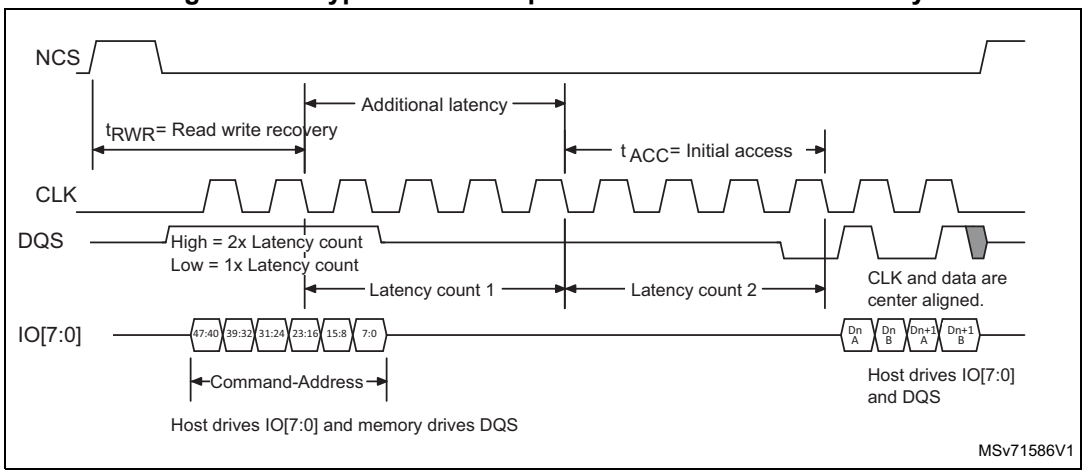


Figure 239. HyperBus write operation with additional latency



Fixed-latency mode

Some devices or some applications may not want to operate with a variable latency time as described above.

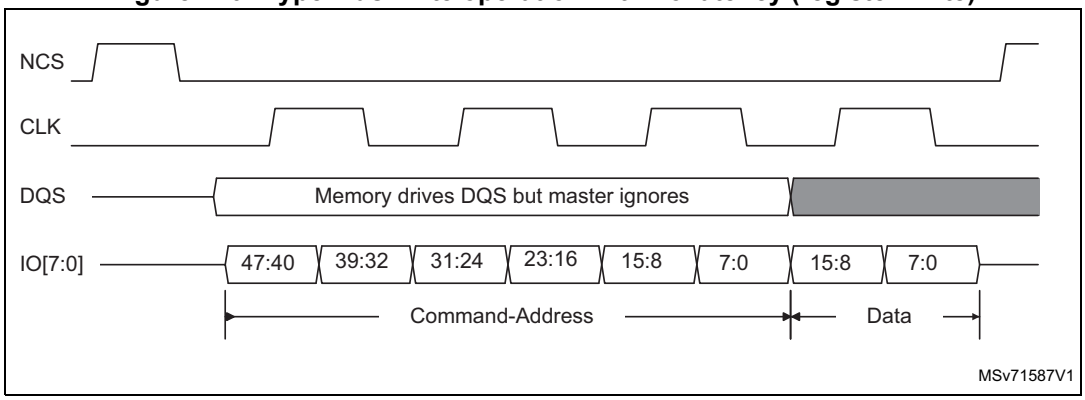
The latency can be forced to 2 x t_{ACC} by setting LM in XSPI_HLCR.

In this XSPI latency mode, the state of the DQS signal is not taken into account by the XSPI, and an additional latency is always added, leading to a fixed 2 x t_{ACC} latency time.

Write operation with no latency

Some devices can also require a zero latency for the write operations. This write-zero latency can be forced by setting WZL in XSPI_HLCR.

Figure 240. HyperBus write operation with no latency (register write)

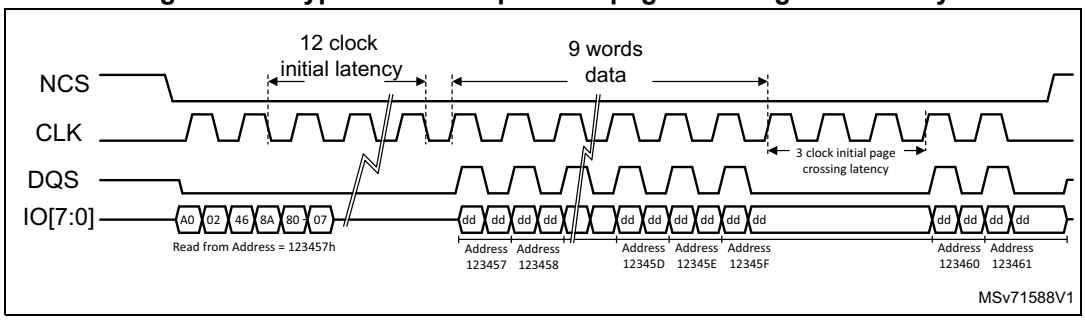


Latency on page-crossing during the read operations

An additional latency can be needed by some devices for the read operation when crossing pages.

The initial latency must be respected for any page access, as a consequence, when the first access is close to the page boundary, a latency is automatically added at the page crossing to respect the t_{ACC} time.

Figure 241. HyperBus read operation page crossing with latency



22.4.8 Specific features

The XSPI supports some specific features, such as:

- Wrap support
- NCS boundary and refresh

Wrap support

The XSPI supports a hybrid wrap as defined by the HyperBus protocol. A hybrid wrap is also supported in the regular-command protocol.

In hybrid wrap, the transaction can continue after the initial wrap with an incremental access.

The wrap size supported by the target memory is configured by WRAPSIZE in XSPI_DCR2.

Wrap is supported only in memory-read direction and only for data size = 8 bytes. Wrapped reads are supported for both HyperBus and regular-command protocols. To enable wrapped-read accesses, the dedicated XSPI_WPxxx registers must be programmed according to the wrapped-read access characteristics. These registers apply for both HyperBus and regular-command protocols.

If the target memory is not supporting the hybrid wrap, WRAPSIZE must be set to 0.

Note: Hybrid wrap requires that the nonwrapped registers (XSPI_CCR, XSPI_TCR, XSPI_IR) are set according to the memory configuration to satisfy its correct data prefetch (initiated after the wrap command).

The wrap operation cannot be interrupted by a refresh. The refresh event is only considered after the wrap completion.

NCS boundary and refresh

Two processes can be activated to regulate the XSPI transactions:

- NCS boundary
- Refresh

The NCS boundary feature limits a transaction to a boundary of aligned addresses. The size of the address to be aligned with, is configured by CSBOUND[4:0] in XSPI_DCR3: it is equal to $2^{CSBOUND}$.

As an example, if CSBOUND(4:0] = 0x4, the boundary is set to 2^4 = 16 bytes. The NCS is then released each time the LSB address is equal to 0xF, and each time a new transaction is issued to address the next data.

If CSBOUND[4:0] = 0, the feature is disabled. A minimum value of three is recommended.

The NCS boundary feature cannot be used for flash memory devices in write mode since a command is necessary to program another page of the flash memory.

The refresh feature limits the duration of the transactions to the value programmed by REFRESH[31:0] in XSPI_DCR4. The duration is expressed in number of cycles. This allows an external RAM to perform its internal refresh operation regularly.

The refresh value must be greater than the minimal transaction size in terms of number of cycles including the command/address/alternate/dummy phases.

If NCS boundary and refresh are enabled at the same time, the NCS is released on the first condition met.

22.4.9 XSPI operating mode introduction

The XSPI has the following operating modes regardless of the low-level protocol used (either regular-command or HyperBus):

- indirect mode (read or write)
- automatic status-polling mode (only in regular-command protocol)
- memory-mapped mode

22.4.10 XSPI indirect mode

In indirect mode, the commands are started by writing to the XSPI registers, and data are transferred by writing or reading the data register, in a similar way to other communication peripherals.

When FMODE[1:0] = 00 in XSPI_CR, the XSPI is in indirect-write mode: bytes are sent to the external device during the data phase. Data are provided by writing to XSPI_DR.

When FMODE[1:0] = 01, the XSPI is in indirect-read mode: bytes are received from the external device during the data phase. Data are recovered by reading XSPI_DR.

In indirect mode, when the XSPI is configured in DTR mode over eight lanes with DQS disabled, the given starting address and the data length must be even.

Note: The XSPI_AR register must be updated even if the start address is the same as the start address of the previous indirect access.

The number of bytes to be read/written is specified in XSPI_DLR:

- If DL[31:0] = 0xFFFF FFFF, the data length is considered undefined and the XSPI simply continues to transfer data until it reaches the end of the external device (as defined by DEVSIZE). If no bytes are to be transferred, DMODE[2:0] must be set to 0 in XSPI_CCR.
- If DL[31:0] = 0xFFFF FFFFF and DEVSIZE[4:0] = 0x1F (its maximum value indicating at 4-Gbyte device), the transfers continue indefinitely, stopping only after an abort request or after the XSPI is disabled. After the last memory address is read (at address 0xFFFF FFFF), reading continues with address = 0x0000 0000.

When the programmed number of bytes to be transmitted or received is reached, the TCF bit is set in XSPI_SR, and an interrupt is generated if TCIE = 1 in XSPI_CR. In the case of an undefined number of data, TCF is set when the limit of the external SPI memory is reached, according to the device size defined in XSPI_DCR1.

Triggering the start of a transfer in regular-command protocol

Depending on the XSPI configuration, there are three different ways to trigger the start of a transfer in indirect mode when using the regular-command protocol. In general, the start of transfer is triggered as soon as the software gives the last information that is necessary for the command. More specifically in indirect mode, a transfer starts when one of the following sequence of events occurs:

- if no address is necessary (ADMODE[2:0] = 000) and if no data need to be provided by the software (FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to INSTRUCTION[31:0] in XSPI_IR
- if an address is necessary (when ADMODE[2:0] ≠ 000) and if no data need to be provided by the software (when FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to ADDRESS[31:0] in XSPI_AR
- if data need to be provided by the software (when FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), and at the moment when a write is performed to DATA[31:0] in XSPI_DR

A write to XSPI_ABR never triggers the communication start. If alternate bytes are required, they must have been programmed before.

As soon as a command is started, the BUSY bit is automatically set in XSPI_SR.

Triggering the start of a transfer in HyperBus protocol

Depending on the XSPI configuration, there are different ways to trigger the start of a command in indirect mode. In general, it is triggered as soon as the firmware gives the last information that is necessary for the transfer to start, and more specifically, a communication in indirect mode is triggered by one of the following register settings, when it is the last one to be executed:

- when a write is performed to ADDRESS[31:0] (XSPI_AR) with ADMODE[2:0] ≠ 000 in indirect read mode (FMODE[1:0] = 01).
- when a write is performed to DATA[31:0] (XSPI_DR) in indirect-write mode (when FMODE = 00).
- when a (dummy) write is performed to INSTRUCTION[31:0] (XSPI_IR) for indirect read mode (with ADMODE[2:0] = 000 and FMODE = 01).

As soon as a transfer is started, the BUSY bit (XSPI_SR[5]) is automatically set.

FIFO and data management

Data in indirect mode pass through a 32-byte FIFO that is internal to the XSPI. FLEVEL in XSPI_SR indicates how many bytes are currently being held in the FIFO.

In indirect-write mode (FMODE[1:0] = 00), the software adds data to the FIFO when it writes in XSPI_DR. A word write adds 4 bytes to the FIFO, a half-word write adds 2 bytes, and a byte write adds only 1 byte. If the software adds too many bytes to the FIFO (more than indicated in DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

The byte/half-word accesses to XSPI_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES is used to define a FIFO threshold after which point the FIFO threshold flag, FTF, gets set. In indirect-read mode, FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. FTF is also set if there is any data left in the FIFO after the

last byte is read from the external device, regardless of FTHRES setting. In indirect-write mode, the FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when the FTF is set. If DMAEN = 1, a DMA transfer is initiated when the FTF is set. The FTF is cleared by hardware as soon as the threshold condition is no longer true (after enough data has been transferred by the CPU or DMA).

The last data read in RX FIFO remains valid as long as there is no request for the next line. This means that, when the application reads several times in a row at the same location, the data is provided from the RX FIFO and not read again from the distant memory.

22.4.11 XSPI automatic status-polling mode

In automatic status-polling mode, the XSPI periodically starts a command to read a defined number of status bytes (up to four). The received bytes can be masked to isolate some status bits and an interrupt can be generated when the selected bits have a defined value. The automatic status-polling mode must be used only in regular-command protocol. For HyperBus protocol, it is not exploitable since the read status register into the HyperFlash memory must be performed in two steps (a write operation followed by a read operation).

The access to the device begins in the same manner as in indirect-read mode. BUSY in XSPI_SR goes high at this point and stays high even between the periodic accesses.

The content of MASK[31:0] in XSPI_PSMAR is used to mask the data from the external device in automatic status-polling mode:

- If the MASK[n] = 0, then bit n of the result is masked and not considered.
- If MASK[n] = 1, and the content of bit[n] is the same as MATCH[n] in XSPI_PSMAR, then there is a match for bit n.

If PMM = 0 in XSPI_CR, the AND-match mode is activated: SMF is set in XSPI_SR only when there is a match on all of the unmasked bits.

If PMM = 1 in XSPI_CR, the OR-match mode is activated: SMF gets set if there is a match on any of the unmasked bits.

An interrupt is called when SMF = 1 if SMIE = 1.

If APMS is set in XSPI_CR, the operation stops and BUSY goes to 0 as soon as a match is detected. Otherwise, BUSY stays at 1 and the periodic accesses continue until there is an abort or until the XSPI is disabled (EN = 0).

XSPI_DR contains the latest received status bytes (FIFO deactivated). The content of this register is not affected by the masking used in the matching logic. FTF in XSPI_SR is set as soon as a new reading of the status is complete. FTF is cleared as soon as the data is read.

In automatic status-polling mode, variable latency is not supported. As a consequence, the memory must then be configured in fixed latency.

22.4.12 XSPI memory-mapped mode

When configured in memory-mapped mode, the external SPI device is seen as an internal memory.

Note: No more than 256 Mbytes can be addressed even if the external device capacity is larger.

If an access is made to an address outside of the range defined by DEVSIZE[4:0] but still within the 256-Mbyte range, then an AXI error is given. The effect of this error depends on the AXI master that attempted the access:

- If it is the Cortex CPU, a hard-fault interrupt is generated.
- If it is a DMA, a DMA transfer error is generated, and the corresponding DMA channel is automatically disabled.

Byte, half-word, and word access types are all supported.

A support for execute in place (XIP) operation is implemented, where the XSPI continues to load the bytes to the addresses following the most recent access. If subsequent accesses are continuous to the bytes that follow, then these operations end up quickly since their results were prefetched.

By default, the XSPI never stops its prefetch operation. It either keeps the previous read operation active with the NCS maintained low or it relaunches a new transfer, even if no access to the external device occurs for a long time. This prefetch feature can be disabled by setting the NOPREF bit in XSPI_DCR1. Disabling the prefetch can be beneficial for performance when the transfer requests are short and jumping frequently between non-consecutive addresses.

Since external devices tend to consume more power when the NCS is held low, the application may want to activate the timeout counter (TCEN = 1 in XSPI_CR): the NCS is released after a period defined by TIMEOUT[15:0] in XSPI_LPTR, when x cycles have elapsed without access since the clock is inactive.

BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is an abort, or the peripheral is disabled.

It is not recommended to program the flash memory using the memory-mapped writes. The indirect-write mode fulfills this operation.

However, if the application requires the use of the MCE for encryption (check MCE product availability), the memory-mapped write mode may be used to program encrypted data to external flash memory under the following conditions:

- Prefetch must be enabled.
- In block cipher mode, the CPU must write a complete 128-bit data block to prevent the MCE from initiating read-modify-write operations when only a few bytes need to be programmed. This precaution avoids incorrect programming operations. There are no specific constraints to respect if the MCE is used in stream cipher mode.
- Apply the abort sequence to exit memory-mapped mode when the data linked to the page has been written in the external memory buffers. The abort sequence triggers the start of the page programming.
- Switch to the automatic status-polling mode to monitor the completion of the page programing phase.
- Relaunch the write enable command in indirect mode, then switch back to the memory-mapped mode configuration to continue to program additional pages if any.

It is recommended to add a synchronization barrier between the end of the controller registers configuration and the first memory-mapped access to the external memory when the controller is configured in memory-mapped mode.

22.4.13 XSPI configuration introduction

The XSPI configuration is done in three steps:

1. XSPI system configuration
2. XSPI device configuration
3. XSPI mode configuration

22.4.14 XSPI system configuration

The XSPI is configured using XSPI_CR. The user must program:

- the functional mode with FMODE[1:0]
- the automatic status-polling mode behavior if needed with PMM and APMS
- the FIFO level with FTHRES
- the DMA use with DMAEN
- the timeout counter use with TCEN
- the dual-memory configuration, if needed, with DMM

In case of an interrupt use, the respective enable bit can also be set during this phase.

If the timeout counter is used, the timeout value is programmed in XSPI_LPTR.

The DMA channel must not be enabled during the XSPI configuration: it must be enabled only when the operation is fully configured, to avoid any unexpected request generation.

The DMA and XSPI must be configured in a coherent manner regarding data length: FTHRES value must reflect the DMA burst size.

22.4.15 XSPI device configuration

The parameters related to the external device targeted are configured through XSPI_DCR1 and XSPI_DCR2.The user must program:

- the device size with DEVSIZE[4:0]
- the chip-select minimum high time with CSHT[5:0]
- the clock mode with FRCK and CKMODE
- the associated delay block DLYB connected externally to the peripheral
- the device frequency with PRESCALER[7:0]

DEVSIZE[4:0] defines the size of external memory using the following formula:

$$\text{Number of bytes in the device} = 2^{[\text{DEVSIZE}+1]}$$

where DEVSIZE+1 is the number of address bits required to address the external device. The external device capacity can go up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

If DMM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

When the XSPI executes two commands, one immediately after the other, it raises the chip-select signal (NCS) high between the two commands for at least one CLK cycle by default.

If the external device requires more time between commands, CSHT[5:0] can be used to specify the minimum number of CLK cycles for which the NCS must remain high.

CKMODE indicates the level that the CLK takes between commands (when NCS = 1).

In HyperBus protocol, the device timing (t_{ACC} and t_{RWR}) and the latency mode must be configured in XSPI_HLCR.

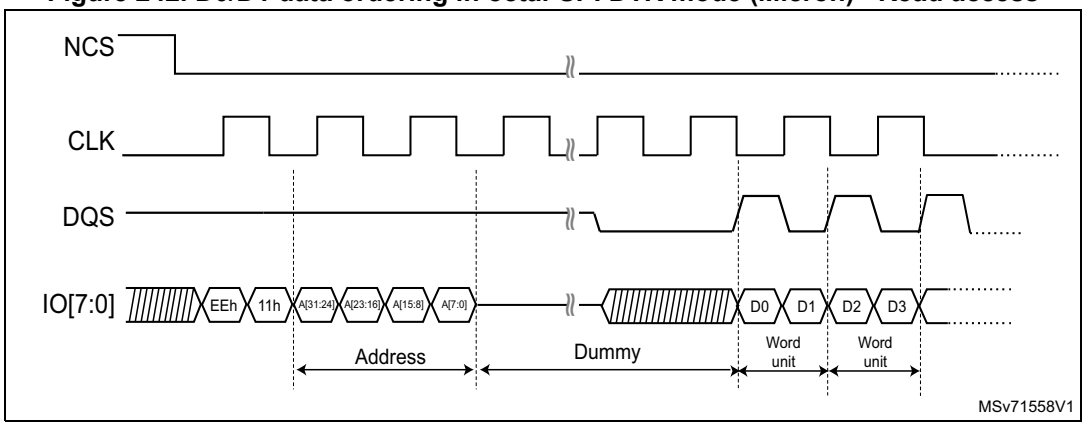
Memory types

External memory providers may present some architecture and slight data formatting differences between them. The bitfield MTYP[2:0] into the XSPI_CR register allows targeting the right controller configuration depending on the associated memory type selected in the application. This is the responsibility of the software developer to align the controller configuration to fit with the targeted memory type.

The memory types are grouped in a such way:

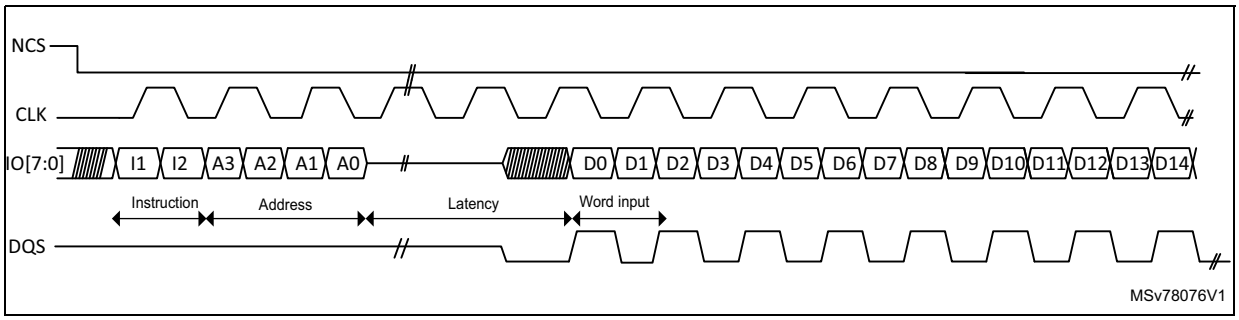
- D0/D1 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 000. For instance, Micron is using such data ordering. In this configuration, the DQS is sent with a polarity inverted respect to the clock polarity.

Figure 242. D0/D1 data ordering in octal-SPI DTR mode (Micron) - Read access



- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 001. For instance, Macronix is using this reverse data ordering in its Octaflash portfolio (this configuration is not adapted to its OctaRAM™ memories). DQS is keeping the same polarity as the clock when reading data from the memory. Refer to *Figure 233: DTR read in octal-SPI mode with DQS (Macronix mode) example*.
- D0/D1 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 010. DQS is keeping the same polarity as the clock when reading data from the memory. It is defined as the standard mode since it is the configuration to select when the targeted memory does not fit with any others modes proposed by the MTYP configuration values.

Figure 243. D0/D1 data ordering in octal-SPI DTR mode with DQS and CLK polarity aligned during Read access



- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 011 with specific address phase built with row and column to fit with Macronix OctaRAM™ memories requirement (refer to *Table 203: OctaRAM command address bit assignment (based on 64 Mb OctaRAM)*. This is the controller which translates internally the targeted address provided by the software in row/column address formatting to sent to the memory. DQS is keeping the same polarity as the clock one when reading data from the memory.

Figure 244. OctaRAM read operation with reverse data ordering D1/D0

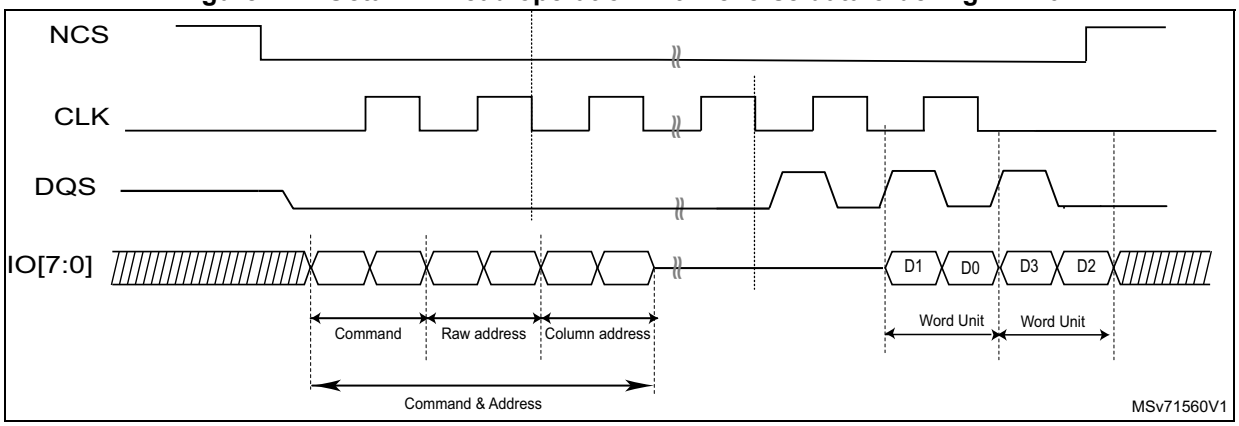


Table 203. OctaRAM command address bit assignment (based on 64 Mb(1) OctaRAM)

Clock	1st clock	2nd clock		3rd clock	
Function	Command	Row address		Column address	
SIO[7]	Command	Reserved	RA7	CA9	Reserved
SIO[6]		Reserved	RA6	CA8	Reserved
SIO[5]		Reserved	RA5	CA7	Reserved
SIO[4]		RA12	RA4	CA6	Reserved
SIO[3]		RA11	RA3	CA5	CA3
SIO[2]		RA10	RA2	CA4	CA2
SIO[1]		RA9	RA1	Reserved	CA1
SIO[0]		RA8	RA0	Reserved	CA0(2)

1. Example of 64 Mb OctaRAM address assignment:
   Row Address [RA12:RA0]: 8K. Column Address [CA9:CA0]: 1K. 64 Mb density = 8K x 1K x 8 bits
2. Column address A0 must be always 0.

- HyperBus memories need to be selected when targeted by the application. The configuration to set depends on the access type:
  - HyperBus memory mode: The protocol follows the HyperBus specification. MTYP[2:0] = 100 is the configuration to use to access the memory space.
  - HyperBus register mode (addressing register space): the memory-mapped accesses in this mode must be noncacheable, or the indirect read/write modes must be used. The configuration to be set for this particular register space access is MTYP[2:0] = 101.

22.4.16 XSPI regular-command mode configuration

Indirect mode configuration

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected, and data can be read from the external device.

When the XSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in XSPI_DLR.
2. Specify the frame timing in XSPI_TCR.
3. Specify the frame format in XSPI_CCR.
4. Specify the instruction in XSPI_IR.
5. Specify the optional alternate byte to be sent right after the address phase in XSPI_ABR.
6. Specify the targeted address in XSPI_AR.
7. Enable the DMA channel if needed.
8. Read/write the data from/to the FIFO through XSPI_DR (if no DMA usage).

If neither the address register (XSPI_AR) nor the data register (XSPI_DR) need to be updated for a particular command, then the command sequence starts as soon as XSPI_IR

is written. This is the case when both ADMODE[2:0] and DMODE[2:0] equal 000, or if just ADMODE[2:0] = 000 when in indirect-read mode (FMODE[1:0] = 01).

When an address is required (ADMODE[2:0] ≠ 000) and the data register does not need to be written (FMODE[1:0] = 01 or DMODE[2:0] = 000), the command sequence starts as soon as the address is updated with a write to XSPI_AR.

In case of data transmission (FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), the communication start is triggered by a write in the FIFO through XSPI_DR.

Automatic status-polling mode configuration

The automatic status-polling mode is enabled by setting FMODE[1:0] = 10. In this mode, the programmed frame is sent and data are retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in XSPI_DLR, it is ignored, and only 4 bytes are read. The periodicity is specified in XSPI_PIR.

Once the status data has been retrieved, the following can be processed:

- Set SMF (an interrupt is generated if enabled).
- Stop automatically the periodic retrieving of the status bytes.

The received value can be masked with the value stored in XSPI_PSMKR, and can be ORed or ANDed with the value stored in XSPI_PSMAR.

In case of a match, SMF is set and an interrupt is generated if enabled. The XSPI can be automatically stopped if AMPS is set. In any case, the latest retrieved value is available in XSPI_DR.

When the XSPI is used in automatic status-polling mode, the frames are constructed in the following way:

1. Specify the input mask in XSPI_PSMKR.
2. Specify the comparison value in XSPI_PSMAR.
3. Specify the read period in XSPI_PIR.
4. Specify a number of data bytes to read in XSPI_DLR.
5. Specify the frame timing in XSPI_TCR.
6. Specify the frame format in XSPI_CCR.
7. Specify the instruction in XSPI_IR.
8. Specify the optional alternate byte to be sent right after the address phase in XSPI_ABR.
9. Specify the optional targeted address in XSPI_AR.

If the address register (XSPI_AR) does not need to be updated for a particular command, then the command sequence starts as soon as XSPI_CCR is written. This is the case when ADMODE[2:0] = 000.

When an address is required (ADMODE[2:0] ≠ 000), the command sequence starts as soon as the address is updated with a write to XSPI_AR.

Memory-mapped mode configuration

In memory-mapped mode, the external device is seen as an internal memory but with some latency during accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using memory-mapped writes, as the internal flags for erase or programming status have to be polled. The indirect-write mode fulfills this operation, possibly in conjunction with the automatic status-polling mode.

The memory-mapped mode is entered by setting FMODE[1:0] = 11 in XSPI_CR.

The programmed instruction and frame are sent when an AXI master accesses the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to XSPI_DR in this mode returns zero.

The data length register (XSPI_DLR) has no meaning in memory-mapped mode.

When the XSPI is used in memory-mapped mode, the frames are constructed in the following way:

1. Specify the frame timing in XSPI_TCR for read operation.
2. Specify the frame format in XSPI_CCR for read operation.
3. Specify the instruction in XSPI_IR.
4. Specify the optional alternate byte to be sent right after the address phase in XSPI_ABR for read operation.
5. Specify the frame timing in XSPI_WTCR for write operation.
6. Specify the frame format in XSPI_WCCR for write operation.
7. Specify the instruction in XSPI_WIR.
8. Specify the optional alternate byte to be sent right after the address phase in XSPI_WABR for write operation.

All configuration operations must be completed (ensured by checking BUSY = 0) before the first access to the memory area: any register write operation when BUSY = 1 has no effect and is not signaled with an error response. On the first access, the XSPI becomes busy, and no further configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit or to abort by setting the ABORT bit.

Memory-mapped write operations are not supported without write strobe.

XSPI delayed data sampling when no DQS is used

By default, when no DQS is used, the XSPI samples the data driven by the external device one half of a CLK cycle after the external device drives the signal.

In case of any external signal delays, it may be useful to sample the data later. Using SSHIFT in XSPI_TCR, the sampling of the data can be shifted by half of a CLK cycle.

The firmware must clear SSHIFT when the data phase is configured in DTR mode (DDTR = 1).

XSPI delayed data sampling when DQS is used

When external DQS is used as a sampling clock, it is shifted precisely by one quarter of the SPI clock cycle, for all frequencies above freq_min, to compensate the data propagation delay in the “high-speed interface” when the product embeds one.

This shift is performed by an external delay block (DLYB) located outside the XSPI. This external delay block has to be calibrated (by enabling FRCK) before starting the first transfer, and recalibrated regularly. The control of this feature depends on the device

implementation (see the delay block (DLYB) section in this document for more details about DLYB and calibration procedure, and the product datasheet for freq_min value).

In configurations where delay does not need to be compensated, the external delay block can be bypassed by setting DLYBYP in XSPI_DCR1.

22.4.17 XSPI HyperBus protocol configuration

Indirect mode configuration (HyperBus)

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected where data can be read from the external device. ADMODE must be configured with a value different from 000 (for instance ADMODE = 100).

When the XSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in XSPI_DLR.
2. Specify the targeted address in XSPI_AR.
3. Enable the DMA channel if needed.
4. Read/write the data from/to the FIFO through XSPI_DR (if no DMA usage).

In indirect-read mode, the command sequence starts as soon as the address is updated with a write to XSPI_AR.

In indirect-write mode, the communication start is triggered by a write in the FIFO through XSPI_DR.

Memory-mapped mode configuration (HyperBus)

In memory-mapped mode, the external device is seen as an internal memory but with some latency during the accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using the memory-mapped writes: the indirect-write mode fulfills this operation.

The memory-mapped mode is entered by setting FMODE[1:0] = 11. The programmed instruction and frame is sent when an AXI master accesses the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to XSPI_DR in this mode returns zero.

The data length register (XSPI_DLR) has no meaning in memory-mapped mode.

All the configuration operation must be completed before the first access to the memory area. On the first access, the XSPI becomes busy, and no configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit, or to abort by setting the ABORT bit.

22.4.18 XSPI error management

The following errors set the TEF flag in XSPI_SR and generates an interrupt if enabled (TEIE = 1 in XSPI_CR):

- in indirect or automatic status-polling mode, when a wrong address has been programmed in XSPI_AR (according to the device size defined by DEVSIZE[4:0]).

- in indirect mode, if the address plus the data length exceed the device size: TEF is set as soon as the access is triggered.

In memory-mapped mode, the XSPI generates an AXI slave error in the following situations, setting the BERRF flag in XSPI_SR and generating an interrupt if enabled (BERRIE = 1 in XSPI_CR) (if the Bus error bit feature is available in the product implementation table, otherwise the Slave bus error response is generated at system level as transaction response with no dedicated bit into the controller status register):

- The memory-mapped mode is disabled and an AXI read or write request occurs.
- A read or write address exceeds the size of the external memory.
- An abort is received while a read or write burst is ongoing when the abort condition is present at the moment of transfer phases preceding the data phase (In incremental or wrap burst type modes). To avoid the error, refer to *Section 22.4.19: XSPI BUSY and ABORT*.
- The XSPI is disabled while a read or write burst is requested.
- A write wrap burst is received.
- A write request is received while DQSE = 0 in XSPI_WCCR in octal DTR mode, in dual-memory configuration, in HyperBus mode or 16-bit mode.
- A read or write request is received while DMODE[2:0] = 000 (no data phase), except when MTYP[2:0] is HyperBus.
- Illegal access size when wrap read burst. This means that the AXI bus Transfer size (ARSIZE) is different from 8 bytes (only for memory-mapped mode).
- Bit DMM is set in XSPI_CR (dual-memory configuration) and octal mode is set.

22.4.19 XSPI BUSY and ABORT

Once the XSPI starts an operation with the external device, BUSY is automatically set in XSPI_SR.

In indirect mode, BUSY is reset once the XSPI has completed the requested command sequence and the FIFO is empty.

In automatic status-polling mode, BUSY goes low only after the last periodic access is complete, due to a match when APMS = 1 or due to an abort.

After the first access in memory-mapped mode, BUSY goes low only on an abort.

Any operation can be aborted by setting ABORT in XSPI_CR. Once the abort is completed, BUSY and ABORT are automatically reset, and the FIFO is flushed.

Before setting ABORT, the software must ensure that all the current transactions are finished using the synchronization barriers. When DMA is enabled to handle the data read or write operations in XSPI_DR, it is recommended to disable the DMA channel before aborting the SPI.

Note: Some devices may misbehave if a write operation to a status register is aborted.

22.4.20 XSPI reconfiguration or deactivation

Before any XSPI reconfiguration, the software must ensure that all the transactions are completed:

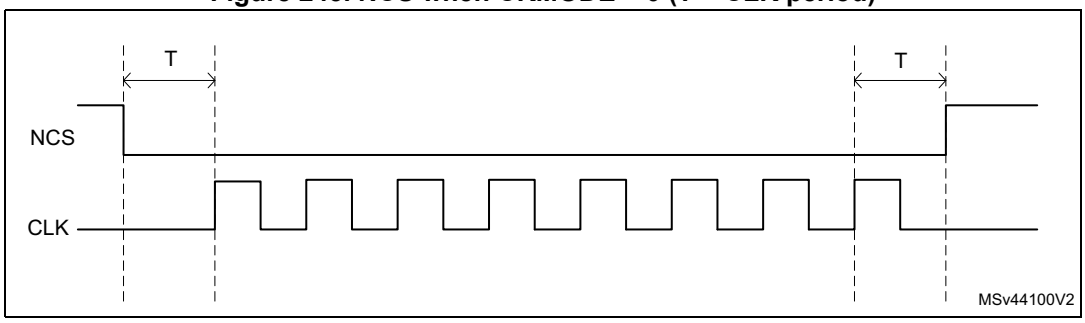
- After a memory-mapped write, the software must perform a dummy read followed by a synchronization barrier, then an abort.
- After a memory-mapped read, the software must perform a synchronization barrier than an abort.

22.4.21 NCS behavior

By default, NCS is high, deselecting the external device. NCS falls before an operation begins and rises as soon as it finishes.

When CKMODE = 0 (clock mode 0: CLK stays low when no operation is in progress), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

Figure 245. NCS when CKMODE = 0 (T = CLK period)



Note: Clock mode 3 is not supported.

In dual-memory configuration, the NCS2 behaves exactly the same as the NCS1. The quad-SPI/octal-SPI memory 2 can then use the NCS1, and the NCS2 pin can be used for other functions.

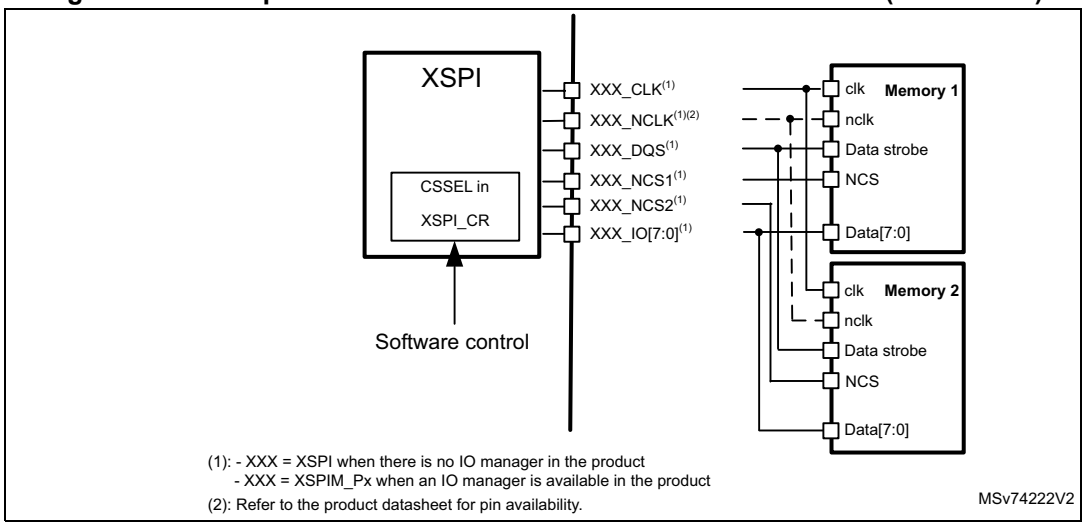
22.4.22 Software control of two external memories

One XSPI instance is able to control the generation of the two chip-select signals (NCS1 and NCS2). This allows two external memories to be connected to the same I/O port driven by the XSPI.

The XSPI generates one NCS signal and the user defines how the NCS is directed to NCS1 or NCS2, by setting the CSSEL bit in XSPI_CR. Intended operation is semistatic commutation, such as starting an application reading an external flash memory and then, for the rest of the application, use an external SRAM connected to the same I/O bus.

From a memory-map point of view, these two memories are located in the same address range.

Figure 246. Example of software control of two external memories (octal mode)



22.4.23 Hardware-controlled extended memory support

One XSPI instance is able to control autonomously the generation of the two chip-select signals (NCS1 and NCS2). This feature is activated by setting EXTENDMEM in XSPI_DCR1 and allows two external memories to be connected to the same I/O port driven by the XSPI.

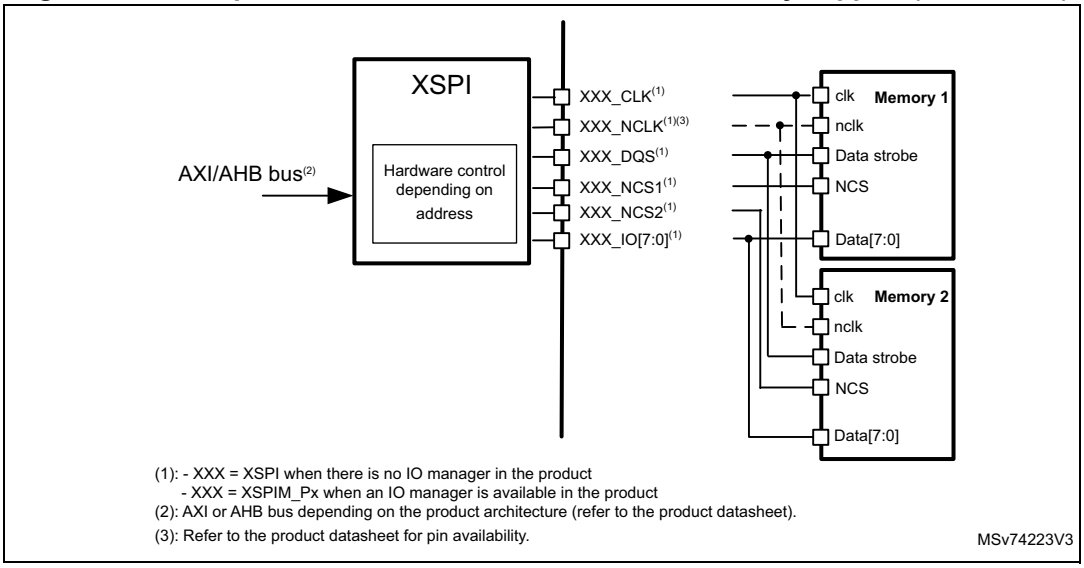
The aim is to extend by a factor two, the available external memory size. Both memories must be contiguous in the memory map. Both external memories must be of the same type and size. The size of one external memory must be set in DEVSIZE of XSPI_CR.

- When the address presented on the interconnect bus in memory-mapped mode or in XSPI_AR in indirect mode is in the range [XSPI_base_address: XSPI_base_address + $2^{DEVSIZE}$ - 1], NCS1 is generated.
- When the address presented on the interconnect bus in memory-mapped mode or in XSPI_AR in indirect mode is in the range [XSPI_base_address + $2^{DEVSIZE}$: XSPI_base_address + 2 × $2^{DEVSIZE}$ - 1], NCS2 is generated.

Note: If the access crosses the middle boundary of the external memory, CS changes automatically (no write into XSPI_AR in indirect mode, or new memory access in memory-mapped mode is needed).

The EXTENDMEM feature (set in bit 21 of XSPI_DCR1) and the Dual-memory mode (set in bit 6 of XSPI_CR) are exclusive: EXTENDMEM = 1 and DMM = 1 as this configuration is not supported.

Figure 247. Example of hardware-controller extended memory support (octal mode)



Note: It is recommended to set EXTENDMEM = 0 for write transfers targeting flash memories (HyperFlash, NOR Flash, etc.).

These memories require preliminary commands that do not include an address phase used by the controller to select the targeted memory.

This recommendation does not apply to RAM memories, as those extra commands are not needed for RAM access.

22.5 Address alignment and data number

The table below summarizes the effect of the address alignment and programmed data number depending on the use case.

Table 204. Address alignment cases

Memory type	Transaction type	Constraint on address(1)	Impact if constraint on address not respected	Constraint on number of bytes(1)	Impact if constraint on bytes not respected
Single, dual, quad flash or SRAM (DMM = 0)	IND(2) read	None	None	None	None
	MM(3) read				
	IND write				
	MM write				
Single, dual, quad flash or SRAM (DMM = 1)	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM write	Even	Slave error	Even	Last byte is lost.

Table 204. Address alignment cases (continued)

Memory type	Transaction type	Constraint on address(1)	Impact if constraint on address not respected	Constraint on number of bytes(1)	Impact if constraint on bytes not respected
Octal flash in SDR mode	IND read	None	None	None	None
	MM read				
	IND write				
	MM write				
Octal memory in DTR mode without WDM(6) or 8-bit HyperFLASH	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM write	Even	Slave error	Even	Last byte is lost.
Octal flash or RAM in DTR mode with WDM	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write(7)				
	MM write				
8-bit HyperRAM	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write(7)				
	MM write				

1. To be respected by the software.
2. IND = indirect mode.
3. MM = memory-mapped mode.
4. Extra data at transfer start.
5. Extra data at transfer end.
6. WDM = write data mask.
7. If the FTHRES bitfield is set to the maximum value with DLR value greater than the data burst length, and if the DMA is enabled or the interrupt based on FIFO THRESHOLD Flag is enabled (FTF), the address must be modulo 2 aligned in DTR mode when the initiator (DMA, CPU, ...) is writing the data with a burst length equal to the FIFO size.

22.6 XSPI interrupts

An interrupt can be produced on the following events:

- Timeout
- Status match
- FIFO threshold
- Transfer complete
- Transfer error

Separate interrupt enable bits are available to provide more flexibility.

Table 205. XSPI interrupt requests

Interrupt event	Event flag	Enable control bit
Timeout	TOF	TOIE
Status match	SMF	SMIE
FIFO threshold	FTF	FTIE
Transfer complete	TCF	TCIE
Transfer error	TEF	TEIE

22.7 XSPI registers

22.7.1 XSPI control register (XSPI_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MSEL	FMODE[1:0]		Res.	NOPREF_AXI	NOPREF	CSSEL	PMM	APMS	BERRIE	TOIE	SMIE	FTIE	TCIE	TEIE
	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FTHRES[4:0]					Res.	DMM	Res.	Res.	TCEN	DMAEN	ABORT	EN
			rw	rw	rw	rw	rw		rw			rw	rw	w	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **MSEL**: External memory select

This bit selects the external memory to be addressed in single-, dual-, quad-SPI mode in single-memory configuration (when DMM = 0).

0: External memory 1 selected (data exchanged over IO[3:0])

1: External memory 2 selected (data exchanged over IO[7:4])

This bit is ignored when DMM = 1 or when octal-SPI mode is selected.

Note: This bitfield can be modified only when BUSY = 0.

Bits 29:28 **FMODE[1:0]**: Functional mode

This bitfield defines the XSPI functional mode of operation.

00: Indirect-write mode

01: Indirect-read mode

10: Automatic status-polling mode (relevant in regular-command protocol only)

11: Memory-mapped mode

If DMAEN = 1 already, then the DMA controller for the corresponding channel must be disabled before changing the FMODE[1:0] value.

Note: This bitfield can be modified only when BUSY = 0.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **NOPREF_AXI**: No prefetch for signaled AXI transactions

This bit disables the automatic prefetch in the external memory when the corresponding AXI transaction is signaled as not-prefetchable.

0: Prefetch is enabled, whatever the AXI prefetch transaction request.

1: Prefetch is disabled when the AXI transaction is signaled as not-prefetchable.

Note: This bit has no effect when NOPREF = 1.

Bit 25 **NOPREF**: No prefetch data

This bit disables the automatic prefetch in the external memory.

0: Automatic prefetch enabled

1: Automatic prefetch disabled

Bit 24 **CSSEL**: Chip-select selection

This bit indicates if it must copy NCS to NCS1 or NCS2.

0: NCS1 active

1: NCS2 active

Note: This bit can be modified only when BUSY = 0.

Bit 23 **PMM**: Polling match mode

This bit indicates which method must be used to determine a match during the automatic status-polling mode.

0: AND-match mode, SMF is set if all the unmasked bits received from the device match the corresponding bits in the match register.

1: OR-match mode, SMF is set if any of the unmasked bits received from the device matches its corresponding bit in the match register.

Note: This bit can be modified only when BUSY = 0.

Bit 22 **APMS**: Automatic status-polling mode stop

This bit determines if the automatic status-polling mode is stopped after a match.

0: Automatic status-polling mode is stopped only by abort or by disabling the XSPI.

1: Automatic status-polling mode stops as soon as there is a match.

Note: This bit can be modified only when BUSY = 0.

Bit 21 **BERRIE**: Bus error interrupt enable

This bit enables the bus error interrupt (if Bus error bit feature is available in the product implementation table).

0: Interrupt disabled

1: Interrupt enabled

Bit 20 **TOIE**: Timeout interrupt enable

This bit enables the timeout interrupt.

0: Interrupt disabled

1: Interrupt enabled

Bit 19 **SMIE**: Status-match interrupt enable

This bit enables the status-match interrupt.

0: Interrupt disabled

1: Interrupt enabled

Bit 18 **FTIE**: FIFO threshold interrupt enable

This bit enables the FIFO threshold interrupt.

0: Interrupt disabled

1: Interrupt enabled

Bit 17 **TCIE**: Transfer complete interrupt enable

This bit enables the transfer complete interrupt.

0: Interrupt disabled

1: Interrupt enabled

Bit 16 **TEIE**: Transfer error interrupt enable

This bit enables the transfer error interrupt.

0: Interrupt disabled

1: Interrupt enabled

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **FTHRES[4:0]**: FIFO threshold level

This bitfield defines, in indirect mode, the threshold number of bytes in the FIFO that causes the FIFO threshold flag FTF in XSPI_SR, to be set.

00000: FTF is set if there are one or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are one or more valid bytes can be read from the FIFO in indirect-read mode.

00001: FTF is set if there are two or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are two or more valid bytes can be read from the FIFO in indirect-read mode.

...

11111: FTF is set if there are 32 free bytes available to be written to in the FIFO in indirect-write mode, or if there are 32 valid bytes can be read from the FIFO in indirect-read mode.

Note: If DMAEN = 1, the DMA controller for the corresponding channel must be disabled before changing the FTHRES[4:0] value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **DMM**: Dual-memory configuration

This bit activates the dual-memory configuration, where two external devices are used simultaneously to double the throughput and the capacity

0: Dual-memory configuration disabled

1: Dual-memory configuration enabled

Note: This bit can be modified only when BUSY = 0.

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **TCEN**: Timeout counter enable

This bit is valid only when the memory-mapped mode (FMODE[1:0] = 11) is selected. This bit enables the timeout counter.

0: The timeout counter is disabled, and thus the chip-select (NCS) remains active indefinitely after an access in memory-mapped mode.

1: The timeout counter is enabled, and thus the chip-select is released in the memory-mapped mode after TIMEOUT[15:0] cycles of external device inactivity.

Note: This bit can be modified only when BUSY = 0.

Bit 2 **DMAEN**: DMA enable

In indirect mode, the DMA can be used to input or output data via XSPI_DR. DMA transfers are initiated when FTF is set.

0: DMA disabled for indirect mode

1: DMA enabled for indirect mode

Note: Resetting the DMAEN bit while a DMA transfer is ongoing, breaks the handshake with the DMA. Do not write this bit during DMA operation.

Bit 1 **ABORT**: Abort request

This bit aborts the ongoing command sequence. This bit stops the current transfer.

0: No abort requested

1: Abort requested

Note: This bit is always read as 0.

Bit 0 **EN**: Enable

This bit enables the XSPI.

0: XSPI disabled

1: XSPI enabled

Note: The DMA request can be aborted without having received the ACK in case this EN bit is cleared during the operation.
In case this bit is set to 0 during a DMA transfer, the REQ signal to DMA returns to inactive state without waiting for the ACK signal from DMA to be active.

22.7.2 XSPI device configuration register 1 (XSPI_DCR1)

Address offset: 0x008

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

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

Bits 26:24 **MTYP[2:0]**: Memory type

This bitfield indicates the type of memory to be supported.

000: Micron mode, D0/D1 ordering in DTR 8-data-bit mode. regular-command protocol in single-, dual-, quad- and octal-SPI modes.

Note: In this mode, DQS signal polarity is inverted with respect to the memory clock signal. This is the default value and care must be taken to change MTYP[2:0] for memories different from Micron.

001: Macronix mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes.

010: Standard mode. D0/D1 ordering with DQS and CLK having the same polarity during read operation. This is the mode to use when the targeted memory does not fit with any others MTYP configuration values. This mode is typically the one targeting the APmemory 8-bit memories for instance.

011: Macronix RAM mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes with dedicated address mapping (address is built with row and column to fit with Macronix requirements).

100: HyperBus memory mode, the protocol follows the HyperBus specification.

101: HyperBus register mode, addressing register space. The memory-mapped accesses in this mode must be noncacheable, or indirect-read/write modes must be used.

Other: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **EXTENDMEM**: Extended memory support

This bit is only relevant when one XSPI drives two same size external memories located in contiguous places in the memory map. In this case, DEVSIZE[4:0] represents the size of each external memory. When this bit is set, the XSPI automatically controls which of NCS1 or NCS2 is active.

0: NCS1 and NCS2 values depend from CSSEL bit in XSPI_CR (software controlled).

1: NCS1 and NCS2 values depend from the address of transfer (hardware controlled).

Note: It is recommended to keep EXTENDMEM = 0 for write operations targeting flash memories (such as NOR Flash and HyperFlash).

Bits 20:16 **DEVSIZE[4:0]**: Device size

This bitfield defines the size of the external device using the following formula:
Number of bytes in device = $2^{[DEVSIZE+1]}$.
DEVSIZE + 1 is effectively the number of address bits required to address the external device. The device capacity can be up to 4 Gbytes (addressed using 32-bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.
In regular-command protocol, if DMM = 1 or EXTENDMEM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

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

Bits 13:8 **CSHT[5:0]**: Chip-select high time

CSHT + 1 defines the minimum number of CLK cycles where the chip-select (NCS) must remain high between commands issued to the external device.

0x0: NCS stays high for at least 1 cycle between external device commands.

0x1: NCS stays high for at least 2 cycles between external device commands.

...

0x3F: NCS stays high for at least 64 cycles between external device commands.

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **DLYBYP**: Delay block bypass

0: The internal sampling clock (called feedback clock) or the DQS data strobe external signal is delayed by the delay block (for more details on this block, refer to the dedicated section of the reference manual as it is not part of the peripheral).

1: The delay block is bypassed, so the internal sampling clock or the DQS data strobe external signal is not affected by the delay block. The delay is shorter than when the delay block is not bypassed, even with the delay value set to minimum value in delay block.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **FRCK**: Free running clock

This bit configures the free running clock.

0: CLK is not free running.

1: CLK is free running (always provided).

Note: Free running clock mode is intended for delay calibration only. No memory or other device access is possible when FRCK is set. Once the bit has been set to 1, the BSY bit is set, the NCS signal remains set to 1 and the clock is sent. The only way to stop the clock is to perform an abort (which clears the BSY bit). Then, the FRCK bit must be cleared by software to allow controller transactions to take place with the external memory.

Bit 0 **CKMODE**: Clock mode 0

This bit is read-only and is read as zero.

22.7.3 XSPI device configuration register 2 (XSPI_DCR2)

Address offset: 0x00C

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **WRAPSIZE[2:0]**: Wrap size

This bitfield indicates the wrap size to which the memory is configured. For memories which have a separate command for wrapped instructions, this bitfield indicates the wrap-size associated with the command held in the XSPI_WPIR register.

000: Wrapped reads are not supported by the memory.

010: External memory supports wrap size of 16 bytes.

011: External memory supports wrap size of 32 bytes.

100: External memory supports wrap size of 64 bytes.

101: External memory supports wrap size of 128 bytes.

Other: Reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **PRESCALER[7:0]**: Clock prescaler

This bitfield defines the scaler factor for generating the CLK based on the kernel clock (value + 1).

0: $F_{CLK} = F_{KERNEL}$, kernel clock used directly as XSPI CLK (prescaler bypassed). In this case, if the DTR mode is used, it is mandatory to provide to the XSPI a kernel clock that has 50% duty-cycle.

1: $F_{CLK} = F_{KERNEL}/2$

2: $F_{CLK} = F_{KERNEL}/3$

...

255: $F_{CLK} = F_{KERNEL}/256$

For odd clock division factors, the CLK duty cycle is not 50 %. The clock signal remains low one cycle longer than it stays high.

22.7.4 XSPI device configuration register 3 (XSPI_DCR3)

Address offset: 0x010

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **CSBOUND[4:0]**: NCS boundary

This bitfield enables the transaction boundary feature. When active, a minimum value of 3 is recommended. The NCS is released on each boundary of $2^{CSBOUND}$ bytes.

0: NCS boundary disabled

Other: NCS boundary set to $2^{CSBOUND}$ bytes

Bits 15:0 Reserved, must be kept at reset value.

22.7.5 XSPI device configuration register 4 (XSPI_DCR4)

Address offset: 0x014

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **REFRESH[31:0]**: Refresh rate

This bitfield enables the refresh rate feature. The NCS is released every REFRESH + 1 clock cycles for writes, and REFRESH + 4 clock cycles for reads. These two values can be extended with few clock cycles when refresh occurs during a byte transmission in single-, dual- or quad-SPI mode, because the byte transmission must be completed.

0: Refresh disabled

Other: Maximum communication length is set to REFRESH + 1 clock cycles.

Note: REFRESH count is based on the memory clock period: if XSPI_DCR2 PRESCALER bitfield is changed, the REFRESH field must be updated accordingly.

22.7.6 XSPI status register (XSPI_SR)

Address offset: 0x020

Reset value: 0x0000 0000

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

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **FLEVEL[5:0]**: FIFO level

This bitfield gives the number of valid bytes that are being held in the FIFO. FLEVEL = 0 when the FIFO is empty, and 32 when it is full.

In automatic status-polling mode, FLEVEL is zero.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **BERRF**: Bus error flag

This bit is set when a bus error occurs. It is cleared by writing 1 to CBERRF. This bit has a meaning if Bus error bit feature is available in the product implementation table.

Bit 5 **BUSY**: Busy

This bit is set when an operation is ongoing. It is cleared automatically when the operation with the external device is finished and the FIFO is empty.

Bit 4 **TOF**: Timeout flag

This bit is set when timeout occurs. It is cleared by writing 1 to CTOF.

Bit 3 **SMF**: Status match flag

This bit is set in automatic status-polling mode when the unmasked received data matches the corresponding bits in the match register (XSPI_PSMAR).

It is cleared by writing 1 to CSMF.

Bit 2 **FTF**: FIFO threshold flag

In indirect mode, this bit is set when the FIFO threshold has been reached, or if there is any data left in the FIFO after the reads from the external device are complete.

It is cleared automatically as soon as the threshold condition is no longer true.

In automatic status-polling mode, this bit is set every time the status register is read, and the bit is cleared when the data register is read.

Bit 1 **TCF**: Transfer complete flag

This bit is set in indirect mode when the programmed number of data has been transferred or in any mode when the transfer has been aborted.It is cleared by writing 1 to CTCF.

Bit 0 **TEF**: Transfer error flag

This bit is set in indirect mode when an invalid address is being accessed in indirect mode.

It is cleared by writing 1 to CTEF.

22.7.7 XSPI flag clear register (XSPI_FCR)

Address offset: 0x024

Reset value: 0x0000 0000

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CBERRF**: Clear bus error flag
Writing 1 clears the BERRF flag in the XSPI_SR register (if Bus error bit feature is available in the product implementation table).

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CTOF**: Clear timeout flag
Writing 1 clears the TOF flag in the XSPI_SR register.

Bit 3 **CSMF**: Clear status match flag
Writing 1 clears the SMF flag in the XSPI_SR register.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CTCF**: Clear transfer complete flag
Writing 1 clears the TCF flag in the XSPI_SR register.

Bit 0 **CTEF**: Clear transfer error flag
Writing 1 clears the TEF flag in the XSPI_SR register.

22.7.8 XSPI data length register (XSPI_DLR)

Address offset: 0x040

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **DL[31:0]**: Data length

Number of data to be retrieved (value+1) in indirect and automatic status-polling modes. A value not greater than three (indicating 4 bytes) must be used for automatic status-polling mode.

All 1's in indirect mode means undefined length, where XSPI continues until the end of the memory, as defined by DEVSIZE.

0x0000_0000: 1 byte is to be transferred.

0x0000_0001: 2 bytes are to be transferred.

0x0000_0002: 3 bytes are to be transferred.

0x0000_0003: 4 bytes are to be transferred.

...

0xFFFF_FFFD: 4,294,967,294 (4G-2) bytes are to be transferred.

0xFFFF_FFFE: 4,294,967,295 (4G-1) bytes are to be transferred.

0xFFFF_FFFF: undefined length; all bytes, until the end of the external device, (as defined by DEVSIZE) are to be transferred. Continue reading indefinitely if DEVSIZE = 0x1F.

DL[0] is stuck at 1 in dual-memory configuration (DMM = 1) even when 0 is written to this bit, thus assuring that each access transfers an even number of bytes.

This bitfield has no effect in memory-mapped mode.

22.7.9 XSPI address register (XSPI_AR)

Address offset: 0x048

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0 and FMODE ≠ 11.

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

Bits 31:0 **ADDRESS[31:0]**: Address

Address to be sent to the external device. In dual-memory configuration, AR[0] is forced to 0.

Caution: Some memory specifications consider that each address corresponds to a 16-bit value. XSPI considers that each address corresponds to an 8-bit value. So the software needs to multiply the address by two when accessing the memory registers.

22.7.10 XSPI data register (XSPI_DR)

Address offset: 0x050

Reset value: 0x0000 0000

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

Bits 31:0 **DATA[31:0]**: Data

Data to be sent/received to/from the external SPI device

In indirect-write mode, data written to this register is stored on the FIFO before it is sent to the external device during the data phase. If the FIFO is too full, a write operation is stalled until the FIFO has enough space to accept the amount of data being written.

In indirect-read mode, reading this register gives (via the FIFO) the data that was received from the external device. If the FIFO does not have as many bytes as requested by the read operation and if BUSY = 1, the read operation is stalled until enough data is present or until the transfer is complete, whichever happens first.

In automatic status-polling mode, this register contains the last data read from the external device (without masking).

Word, half-word, and byte accesses to this register are supported. In indirect-write mode, a byte write adds 1 byte to the FIFO, a half-word write 2 bytes, and a word write 4 bytes.

Similarly, in indirect-read mode, a byte read removes 1 byte from the FIFO, a halfword read 2 bytes, and a word read 4 bytes. Accesses in indirect mode must be aligned to the bottom of this register: A byte read must read DATA[7:0] and a half-word read must read DATA[15:0].

22.7.11 XSPI polling status mask register (XSPI_PSMKR)

Address offset: 0x080

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **MASK[31:0]**: Status mask

Mask to be applied to the status bytes received in automatic status-polling mode

For bit n:

0: Bit n of the data received in automatic status-polling mode is masked and its value is not considered in the matching logic.

1: Bit n of the data received in automatic status-polling mode is unmasked and its value is considered in the matching logic.

22.7.12 XSPI polling status match register (XSPI_PSMAR)

Address offset: 0x088

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **MATCH[31:0]**: Status match
Value to be compared with the masked status register to get a match

22.7.13 XSPI polling interval register (XSPI_PIR)

Address offset: 0x090

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INTERVAL[15:0]**: Polling interval
Number of CLK cycles between a read during the automatic status-polling phases

22.7.14 XSPI communication configuration register (XSPI_CCR)

Address offset: 0x100

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for data phase
1: DTR mode enabled for data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Other: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate- byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Other: Reserved

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

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Other: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for the instruction phase
1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Other: Reserved

22.7.15 XSPI timing configuration register (XSPI_TCR)

Address offset: 0x108

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSHIFT**: Sample shift
By default, the XSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.
This bit allows the data to be sampled later in order to consider the external signal delays.
0: No shift
1: 1/2 cycle shift
The software must ensure that SSHIFT = 0 when the data phase is configured in DTR mode (when DDTR = 1.)

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DHQC**: Delay hold quarter cycle
0: No delay hold
1: 1/4 cycle hold

Bits 27:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

22.7.16 XSPI instruction register (XSPI_IR)

Address offset: 0x110

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

22.7.17 XSPI alternate bytes register (XSPI_ABR)

Address offset: 0x120

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes

Optional data to be sent to the external SPI device right after the address.

22.7.18 XSPI low-power timeout register (XSPI_LPTR)

Address offset: 0x130

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TIMEOUT[15:0]**: Timeout period

After each access in memory-mapped mode, the XSPI prefetches the subsequent bytes and hold them in the FIFO.

This bitfield indicates how many CLK cycles the XSPI waits after the clock becomes inactive and until it raises the NCS, putting the external device in a lower-consumption state.

22.7.19 XSPI wrap communication configuration register (XSPI_WPCCR)

Address offset: 0x140

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable

This bit enables the data strobe management.

0: DQS disabled

1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for the data phase
1: DTR mode enabled for the data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Other: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate-byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: no alternate bytes
001: alternate bytes on a single line
010: alternate bytes on two lines
011: alternate bytes on four lines
100: alternate bytes on eight lines
Other: reserved

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

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for address phase
1: DTR mode enabled for address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Other: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines the instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for the instruction phase
1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Other: Reserved

22.7.20 XSPI wrap timing configuration register (XSPI_WPTCR)

Address offset: 0x148

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSHIFT**: Sample shift
By default, the XSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.
This bit allows the data to be sampled later in order to consider the external signal delays.
0: No shift
1: 1/2 cycle shift
The firmware must assure that SSHIFT=0 when the data phase is configured in DTR mode (when DDTR = 1).

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DHQC**: Delay hold quarter cycle
Add a quarter cycle delay on the outputs in DTR communication to match hold requirement.
0: No quarter cycle delay
1: 1/4 cycle delay inserted

Bits 27:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

22.7.21 XSPI wrap instruction register (XSPI_WPIR)

Address offset: 0x150

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

22.7.22 XSPI wrap alternate bytes register (XSPI_WPABR)

Address offset: 0x160

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes
Optional data to be sent to the external SPI device right after the address

22.7.23 XSPI write communication configuration register (XSPI_WCCR)

Address offset: 0x180

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for the data phase
1: DTR mode enabled for the data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Other: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate bytes double transfer rate
This bit sets the DTR mode for the alternate-bytes phase.
0: DTR mode disabled for alternate-bytes phase
1: DTR mode enabled for alternate-bytes phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Other: Reserved

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

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Other: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines the instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for instruction phase
1: DTR mode enabled for instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Other: Reserved

22.7.24 XSPI write timing configuration register (XSPI_WTCR)

Address offset: 0x188

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

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

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

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency. In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

22.7.25 XSPI write instruction register (XSPI_WIR)

Address offset: 0x190

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

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

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

22.7.26 XSPI write alternate bytes register (XSPI_WABR)

Address offset: 0x1A0

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

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

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes

Optional data to be sent to the external SPI device right after the address

22.7.27 XSPI HyperBus latency configuration register (XSPI_HLCR)

Address offset: 0x200

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **TRWR[7:0]:** Read-write minimum recovery time

Device read-to-write/write-to-read minimum recovery time expressed in number of communication clock cycles

Bits 15:8 **TACC[7:0]**: Access time

Device access time according to the memory latency, expressed in number of communication clock cycles

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

Bit 1 **WZL:** Write zero latency

This bit enables zero latency on write operations.

0: Latency on write accesses

1: No latency on write accesses

Bit 0 **LM:** Latency mode

This bit selects the latency mode.

0: Variable initial latency

1: Fixed latency

Note: This bit must be set when using the dual-octal HyperRAM configuration. It must be kept cleared for HyperFLASH memory.

22.7.28 XSPI register map

Table 206. XSPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	XSPI_CR	Res.	MSEL	FMDOE[1:0]		Res.	NOPREF_AXI	NOPREF	CSSEL	PMM	APMS	BERRIE	TOIE	SMIE	FTIE	TCIE	TEIE	Res.	Res.	Res.	FTHRES[4:0]					Res.	DMM	Res.	Res.	TCEN	DMAEN	ABORT	EN
	Reset value		0	0	0		0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0		0			0	0	0	0
0x004	Reserved																																
0x008	XSPI_DCR1	Res					MTYP [2:0]			Res.	Res.	EXTENDMEM	DEVSIZE[4:0]					Res.	Res.	CSHT[5:0]						Res.	Res.	Res.	Res.	DLYBYP	Res.	FRCK	CKMODE
	Reset value						0	0	0			0	0	0	0	0	0			0	0	0	0	0	0					0		0	0
0x00C	XSPI_DCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRAPSIZE [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[7:0]							
	Reset value														0	0	0									0	0	0	0	0	0	0	0
0x010	XSPI_DCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSBOUND[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												0	0	0	0	0																
0x014	XSPI_DCR4	REFRESH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018-0x01C	Reserved																																
0x020	XSPI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLEVEL[5:0]						Res.	BERRF	BUSY	TOF	SMF	FTF	TCF	TEF
	Reset value																			0	0	0	0	0	0		0	0	0	0	0	0	0
0x024	XSPI_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BERRF	Res.	CTOF	CSMF	Res.	CTCF	CTEF
	Reset value																										0		0	0		0	0
0x028-0x03C	Reserved																																
0x040	XSPI_DLR	DL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	Reserved	Reserved																															

Table 206. XSPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x048	XSPI_AR	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	Reserved																																
0x050	XSPI_DR	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054-0x07C	Reserved																																
0x080	XSPI_PSMKR	MASK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	Reserved																																
0x088	XSPI_PSMAR	MATCH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	Reserved																																
0x090	XSPI_PIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTERVAL[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094-0x0FC	Reserved																																
0x100	XSPI_CCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE [2:0]		
	Reset value			0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x104	Reserved																																
0x108	XSPI_TCR	Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value		0		0																								0	0	0	0	0
0x10C	Reserved																																
0x110	XSPI_IR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x114-0x11C	Reserved																																
0x120	XSPI_ABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x124-0x12C	Reserved																																
0x130	XSPI_LPTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMEOUT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x134-0x13C	Reserved																																

Table 206. XSPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x140	XSPI_WPCCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE [1:0]		IDTR	IMODE [2:0]		
	Reset value			0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x144	Reserved																																
0x148	XSPI_WPTCR	Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value		0		0																								0	0	0	0	0
0x14C	Reserved																																
0x150	XSPI_WPIR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x154-0x15C	Reserved																																
0x160	XSPI_WPABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x164-0x17C	Reserved																																
0x180	XSPI_WCCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE [1:0]		IDTR	IMODE [2:0]		
	Reset value			0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x184	Reserved																																
0x188	XSPI_WTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value																												0	0	0	0	0
0x18C	Reserved																																
0x190	XSPI_WIR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x94-0x019C	Reserved																																
0x1A0	XSPI_WABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1A4-0x1FC	Reserved																																
0x200	XSPI_HLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRWR[7:0]								TACC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	WZL	LM
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

23 Secure digital input/output MultiMediaCard interface (SDMMC)

23.1 SDMMC main features

The SD/SDIO, embedded MultiMediaCard (*e*•MMC) host interface (SDMMC) provides an interface between the AHB bus and SD memory cards, SDIO cards and *e*•MMC devices.

The MultiMediaCard system specifications are available through the MultiMediaCard Association website at www.jedec.org, published by the MMCA technical committee.

SD memory card and SD I/O card system specifications are available through the SD card Association website at www.sdcard.org.

The SDMMC features include the following:

- Compliance with *Embedded MultiMediaCard System Specification Version 5.1*. Card support for three different databus modes: 1-bit (default), 4-bit and 8-bit. (HS200 SDMMC_CK speed limited to maximum allowed I/O speed)(HS400 is not supported).
- Full compatibility with previous versions of MultiMediaCards (backward compatibility).
- Full compliance with *SD memory card specifications version 6.0*. (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Full compliance with *SDIO card specification version 4.0.* Card support for two different databus modes: 1-bit (default) and 4-bit. (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Data transfer up to 208 Mbyte/s for the 8-bit mode. (depending maximum allowed I/O speed).
- Data and command output enable signals to control external bidirectional drivers.
- IDMA linked list support

The MultiMediaCard/SD bus connects cards to the host.

The current version of the SDMMC supports only one SD/SDIO/*e*•MMC card at any one time and a stack of *e*•MMC.

23.2 SDMMC implementation

Table 207. SDMMC features

SDMMC modes/features(1)	SDMMC1
Variable delay	X
SDMMC_CKIN	X
SDMMC_CDIR, SDMMC_D0DIR	X
SDMMC_D123DIR	X

1. X = supported.

23.3 SDMMC bus topology

Communication over the bus is based on command/response and data transfers.

The basic transaction on the SD/SDIO/*e*•MMC bus is the command/response transaction. These types of bus transaction transfer their information directly within the command or response structure. In addition, some operations have a data token.

Data transfers are done in the following ways:

- Block mode: data block(s) with block size 2^N bytes with N in the range 0-14
- SDIO multibyte mode: single data block with block size range 1-512 bytes
- *e*•MMC Stream mode: continuous data stream

Data transfers to/from *e*•MMC cards are done in data blocks or streams.

Figure 248. SDMMC "no response" and "no data" operations

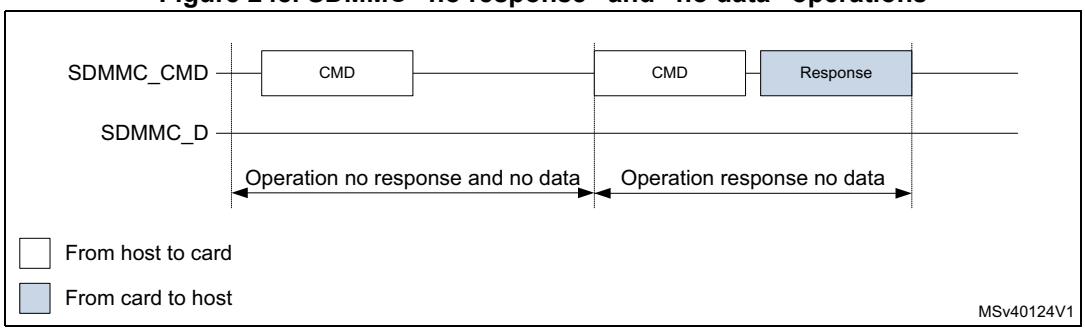
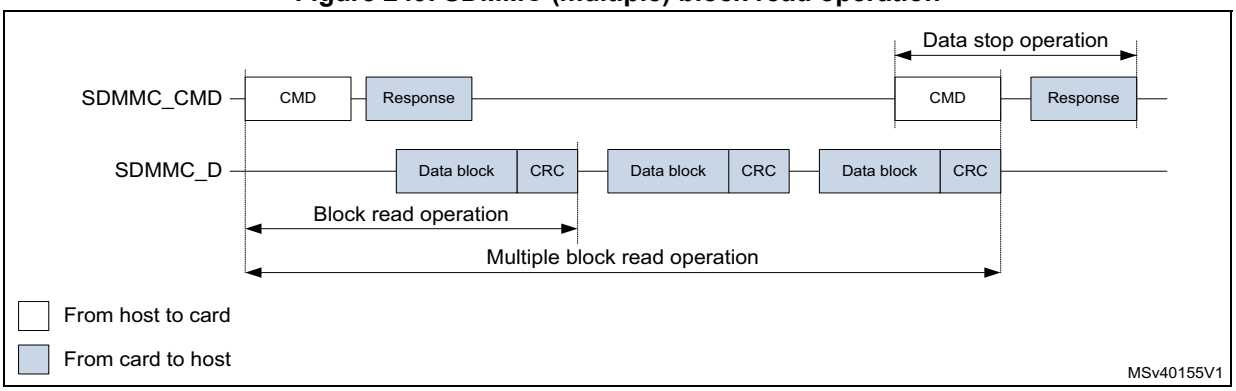
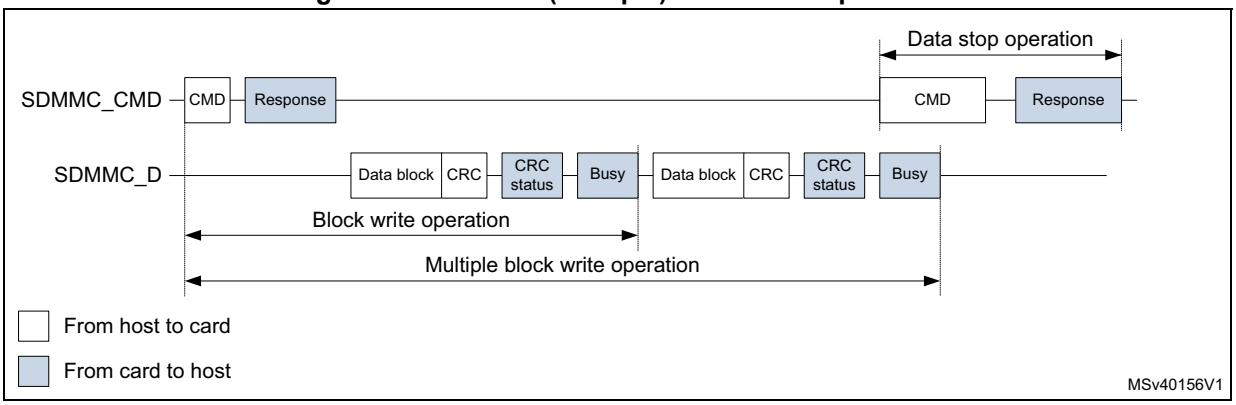


Figure 249. SDMMC (multiple) block read operation



Note: The Stop Transmission command is not required at the end of a e•MMC multiple block read with predefined block count.

Figure 250. SDMMC (multiple) block write operation



Note: The Stop Transmission command is not required at the end of an e•MMC multiple block write with predefined block count.

The SDMMC does not send any data as long as the Busy signal is asserted (SDMMC_D0 pulled low).

Figure 251. SDMMC (sequential) stream read operation

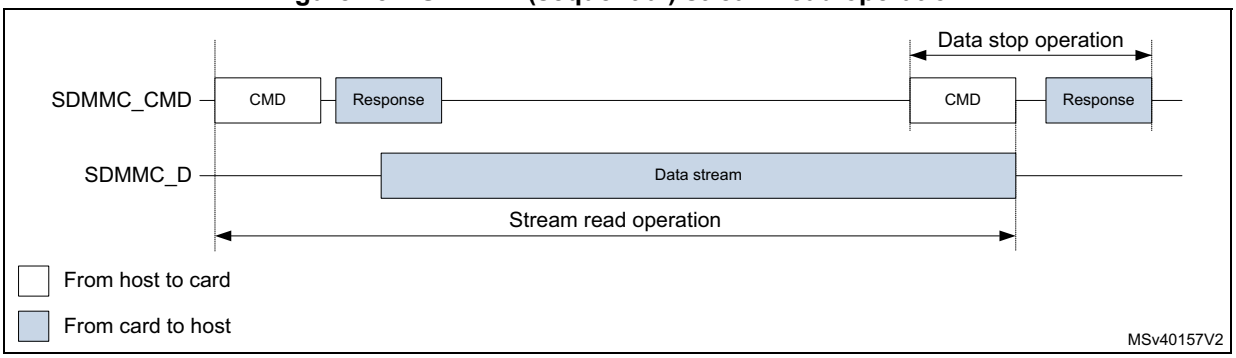
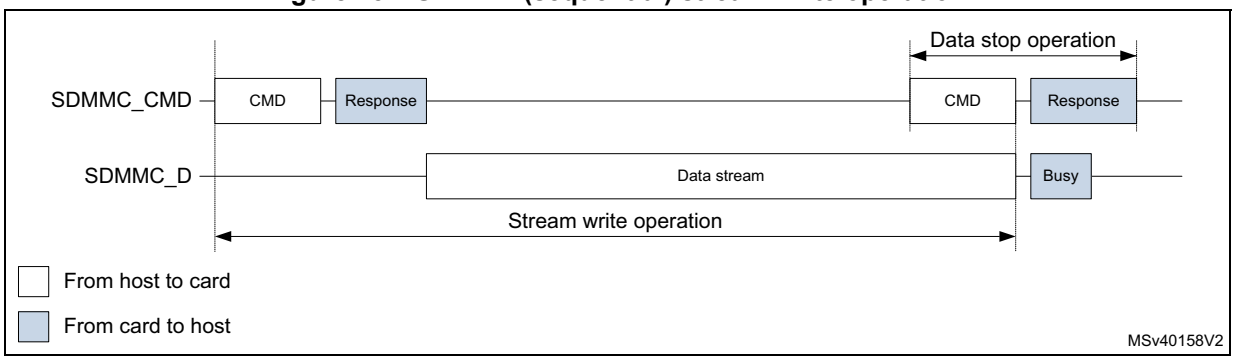


Figure 252. SDMMC (sequential) stream write operation



Stream data transfer operates only in a 1-bit wide bit bus configuration on SDMMC_D0 in single data rate modes (DS, HS, and SDR).

23.4 SDMMC operation modes

Table 208. SDMMC operation modes SD and SDIO

SDIO bus speed modes(1)(2)	Max bus speed (3) [Mbyte/s]	Max clock frequency [MHz](4)	Signal voltage [V]
DS (default speed)	12.5	25	3.3
HS (high speed)	25	50	3.3
SDR12	12.5	25	1.8
SDR25	25	50	1.8
DDR50	50	50	1.8
SDR50	50	100	1.8
SDR104	104	208	1.8

1. SDR single data rate signaling.
2. DDR double data rate signaling (data is sampled on both SDMMC_CK clock edges).
3. SDIO bus speed with 4-bit bus width.
4. Maximum frequency depending on maximum allowed I/O speed.

SDR104 mode requires variable delay support using sampling point tuning. The use of variable delay is optional for SDR50 mode.

Table 209. SDMMC operation modes *e*•MMC

e•MMC bus speed modes (1)(2)	Max bus speed (3) [Mbyte/s]	Max clock frequency [MHz](4)	Signal voltage [V](5)
Legacy compatible	26	26	3/1.8/1.2V
High speed SDR	52	52	3/1.8/1.2V
High speed DDR	104	52	3/1.8/1.2V
High speed HS200	200	200	1.8/1.2V

1. SDR single data rate signaling.
2. DDR double data rate signaling. (data is sampled on both SDMMC_CK clock edges).
3. *e*•MMC bus speed with 8-bit bus width.
4. Maximum frequency depending on maximum allowed I/O speed.
5. Supported signal voltage level depends on I/O port characteristics, refer to device datasheet.

HS200 mode requires variable delay support using sampling point tuning.

23.5 SDMMC functional description

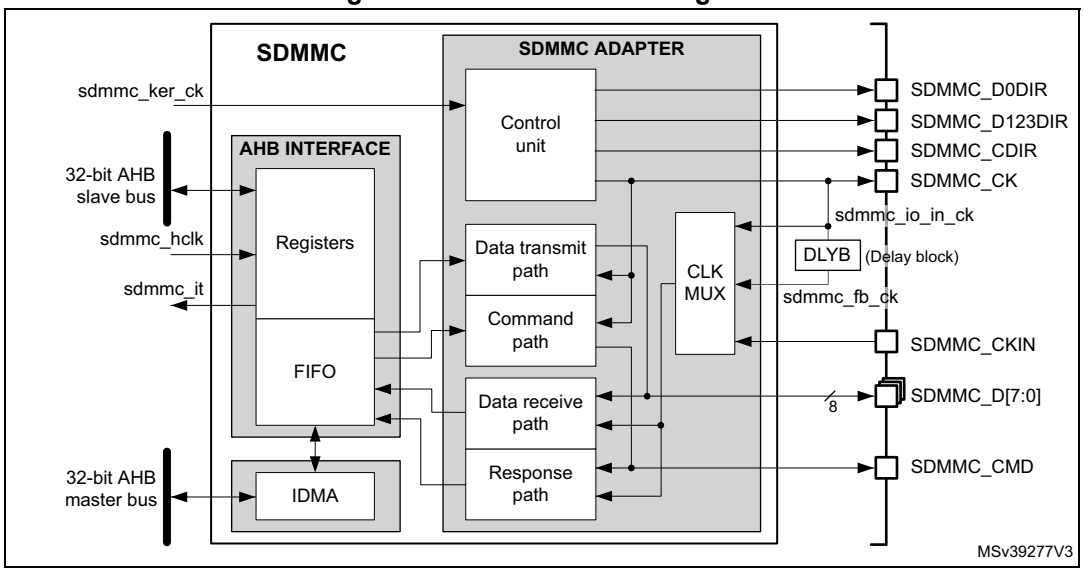
The SDMMC consists of four parts:

- The AHB slave interface accesses the SDMMC adapter registers, and generates interrupt signals and IDMA control signals.
- The SDMMC adapter block provides all functions specific to the *e*•MMC/SD/SD I/O card such as the clock generation unit, command and data transfer.
- The internal DMA (IDMA) block with its AHB master interface.
- A delay block (DLYB) taking care of the receive data sample clock alignment. The delay block is NOT part of the SDMMC. A delay block is mandatory when supporting SDR104 or HS200.

23.5.1 SDMMC block diagram

Figure 253 shows the SDMMC block diagram.

Figure 253. SDMMC block diagram



23.5.2 SDMMC pins and internal signals

Table 210 lists the SDMMC internal input/output signals, *Table 211* the SDMMC pins (alternate functions).

Table 210. SDMMC internal input/output signals

Signal name	Signal type	Description
sdmmc_ker_ck	Digital input	SDMMC kernel clock
sdmmc_hclk	Digital input	AHB clock
sdmmc_it	Digital output	SDMMC global interrupt

Table 210. SDMMC internal input/output signals (continued)

Signal name	Signal type	Description
sdmmc_io_in_ck	Digital input	SD/SDIO/*e*•MMC card feedback clock. This signal is internally connected to the SDMMC_CK pin (for DS and HS modes).
sdmmc_fb_ck	Digital input	SD/SDIO/*e*•MMC card tuned feedback clock after DLYB delay block (for SDR50, DDR50, SDR104, HS200)

Table 211. SDMMC pins

Pin name	Pin type	Description
SDMMC_CK	Digital output	Clock to SD/SDIO/*e*•MMC card
SDMMC_CKIN	Digital input	Clock feedback from an external driver for SD/SDIO/*e*•MMC card. (for SDR12, SDR25, SDR50, DDR50)
SDMMC_CMD	Digital input/output	SD/SDIO/*e*•MMC card bidirectional command/response signal.
SDMMC_CDIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the SDMMC_CMD signal.
SDMMC_D[7:0]	Digital input/output	SD/SDIO/*e*•MMC card bidirectional data lines.
SDMMC_D0DIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the SDMMC_D0 data line.
SDMMC_D123DIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the data lines SDMMC_D[3:1].

23.5.3 General description

The **SDMMC_D[7:0]** lines have different operating modes:

- By default, SDMMC_D0 line is used for data transfer. After initialization, the host can change the databus width.
- For an *e*•MMC, 1-bit (SDMMC_D0), 4-bit (SDMMC_D[3:0]) or 8-bit (SDMMC_D[7:0]) data bus widths can be used.
- For an SD or an SDIO card, 1-bit (SDMMC_D0) or 4-bit (SDMMC_D[3:0]) can be used. All data lines operate in push-pull mode.

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the data lines is indicated with I/O direction signals. The **SDMMC_D0DIR** signal indicates the I/O direction for the SDMMC_D0 data line, the **SDMMC_D123DIR** for the SDMMC_D[3:1] data lines.

SDMMC_CMD only operates in push-pull mode:

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the SDMMC_CMD line is indicated with the I/O direction signal **SDMMC_CDIR**.

SDMMC_CK clock to the card originates from **sdmmc_ker_ck**:

- When the sdmmc_ker_ck clock has 50 % duty cycle, it can be used even in bypass mode (CLKDIV = 0).
- When the sdmmc_ker_ck duty cycle is not 50 %, the CLKDIV must be used to divide it by 2 or more (CLKDIV > 0).
- The phase relation between the SDMMC_CMD / SDMMC_D[7:0] outputs and the SDMMC_CK can be selected through the NEGEDGE bit. The phase relation depends on the CLKDIV, NEGEDGE, and DDR settings. See *Figure 254*.

Figure 254. SDMMC Command and data phase relation

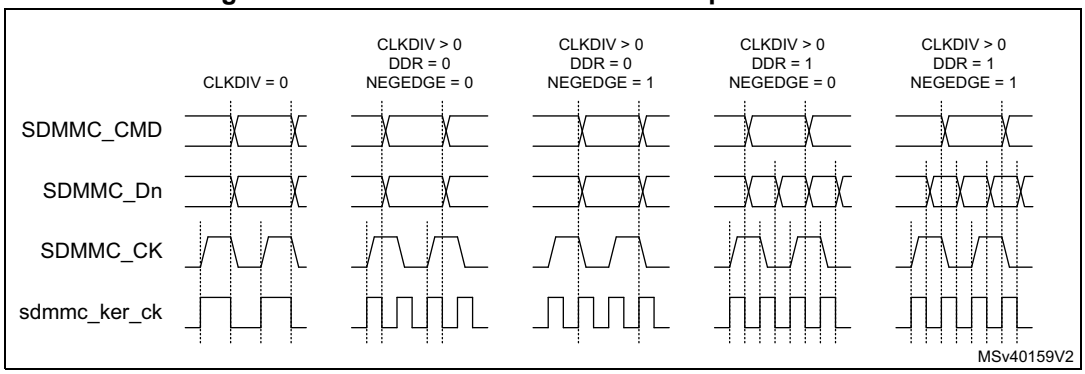


Table 212. SDMMC Command and data phase selection

CLKDIV	DDR	NEGEDGE	SDMMC_CK	Command out	Data out
0	x	x	= sdmmc_ker_ck	Generated on sdmmc_ker_ck falling edge	
>0	0	0	Generated on sdmmc_ker_ck rising edge	Generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.	
		1		Generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.	
	1	0		Generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.	Generated on sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.
		1		Generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.	

By default, the **sdmmc_io_in_ck** feedback clock input is selected for sampling incoming data in the SDMMC receive path. It is derived from the SDMMC_CK pin.

For tuning the phase of the sampling clock to accommodate the receive data timing, the DLYB delay block available on the device can be connected between **sdmmc_io_in_ck** signal (DLYB input dlyb_in_ck) and **sdmmc_fb_ck** clock input of SDMMC (DLYB output dlyb_out_ck). Selecting the **sdmmc_fb_ck** clock input in the receive path then enables using the phase-tuned sampling clock for the incoming data. This is required for SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

When using an external driver (a voltage switch transceiver), the SDMMC_CKIN feedback clock input can be selected to sample the receive data.

For an SD/SDIO/*e*•MMC card, the clock frequency can vary between 0 and 208 MHz (limited by maximum I/O speed).

Depending on the selected bus mode (SDR or DDR), one bit or two bits are transferred on SDMMC_D[7:0] lines with each clock cycle. The SDMMC_CMD line transfers only one bit per clock cycle.

23.5.4 SDMMC adapter

The SDMMC adapter (see *Figure 253: SDMMC block diagram*) is a multimedia/secure digital memory card bus master that provides an interface to a MultiMediaCard stack or to a secure digital memory card. It consists of the following subunits:

- Control unit
- Data transmit path
- Command path
- Data receive path
- Response path
- Receive data path clock multiplexer
- Delay block (DLYB), external to the SDMMC
- Adapter register block
- Data FIFO
- Internal DMA (IDMA)

Note: The adapter registers and FIFO use the AHB clock domain (sdmmc_hclk). The control unit, command path and data transmit path use the SDMMC adapter clock domain (sdmmc_ker_ck). The response path and data receive path use the SDMMC adapter feedback clock domain from the sdmmc_io_in_ck, or SDMMC_CKIN, or from the sdmmc_fb_ck generated by DLYB.

The DLYB delay block on the device can be used in conjunction with the SDMMC adapter, to tune the phase of the sampling clock for incoming data in SDMMC receive mode. It is required for the SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

Adapter register block

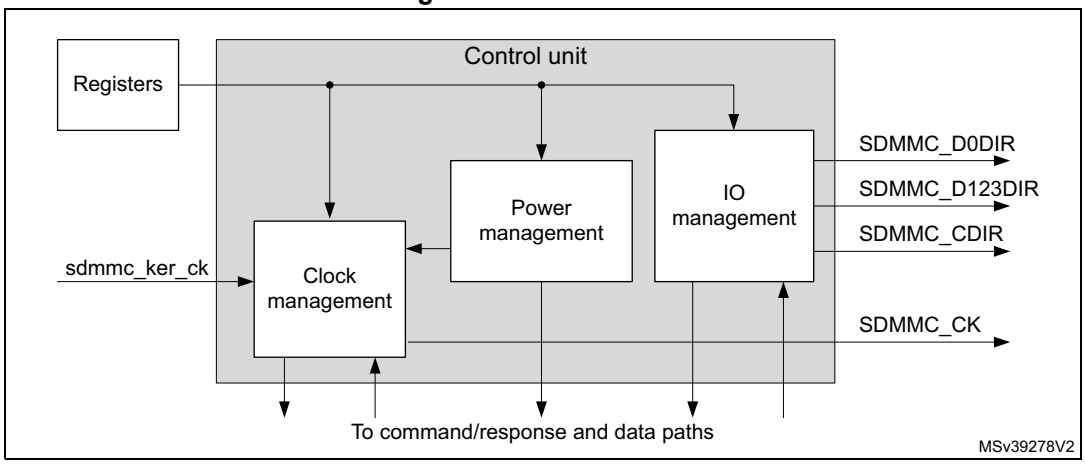
The adapter register block contains all system control registers, the SDMMC command and response registers and the data FIFO.

This block also generates the signals from the corresponding bit location in the SDMMC Clear register that clear the static flags in the SDMMC adapter.

Control unit

The control unit illustrated in *Figure 255*, contains the power management functions, the SDMMC_CK clock management with divider, and the I/O direction management.

Figure 255. Control unit



The power management subunit disables the card bus output signals during the power-off and power-up phases.

There are three power phases:

- power-off
- power-up
- power-on

The clock management subunit uses the sdmmc_ker_ck to generate the SDMMC_CK and provides the division control. It also takes care of stopping the SDMMC_CK for flow control, for example.

The clock outputs are inactive:

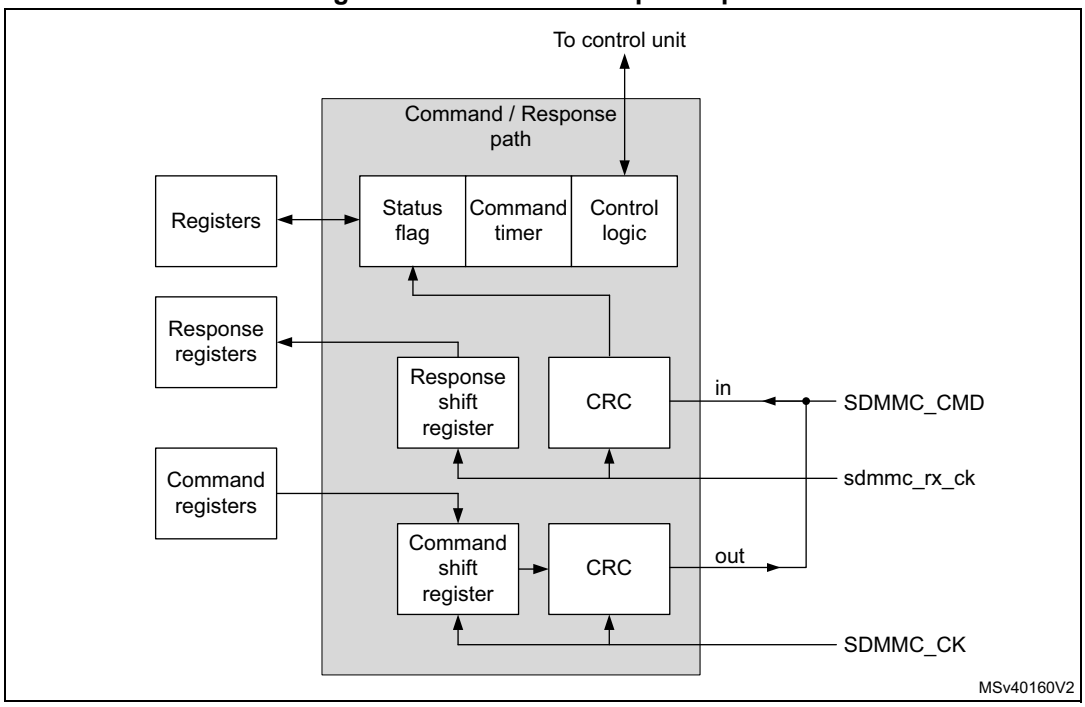
- after reset
- during the power-off or power-up phases
- if the power saving mode (register bit PWRSAV) is enabled and the card bus is in the Idle state for eight clock periods. The clock is stopped eight cycles after both the command/response CPSM and data path DPSM subunits have enter the Idle phase. The clock is restarted when the command/response CPSM or data path DPSM is activated (enabled).

The I/O management subunit takes care of the SDMMC_Dn and SDMMC_CMD I/O direction signals, which controls the external voltage transceiver.

Command/response path

The command/response path subunit transfers commands and responses on the SDMMC_CMD line. The command path is clocked on the SDMMC_CK and sends commands to the card,. The response path is clocked on the sdmmc_rx_ck and receives responses from the card.

Figure 256. Command/response path

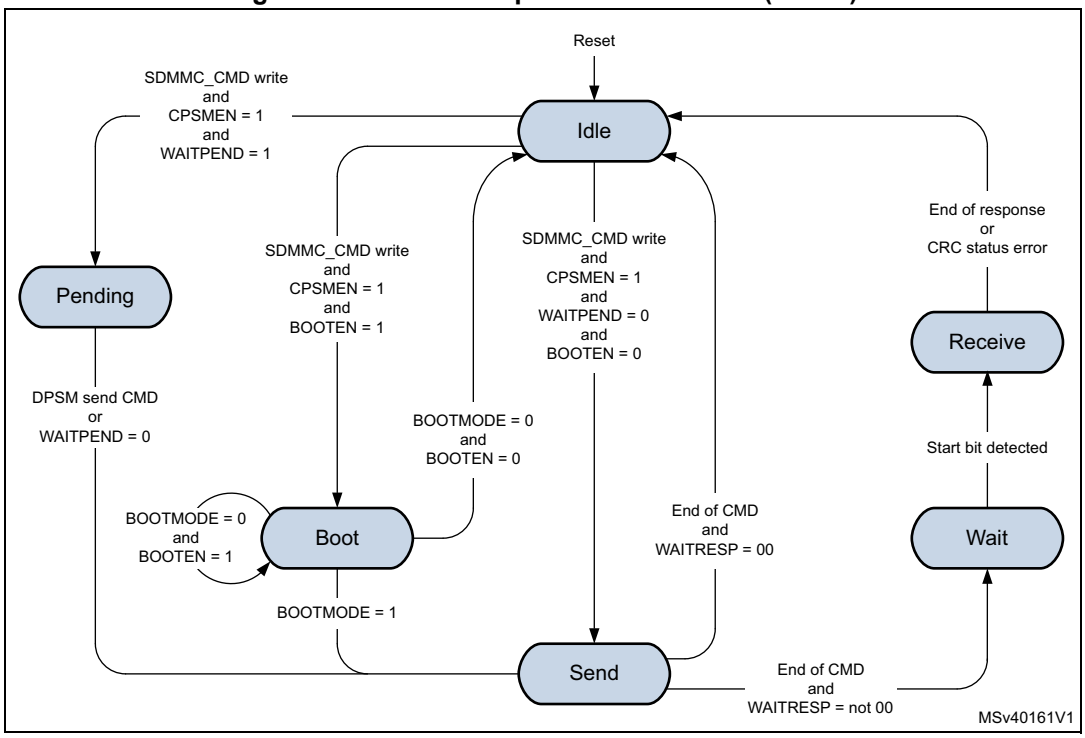


Command/response path state machine (CPSM):

- When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent the CRC is appended and the command path state machine (CPSM) sets the status flags and:
  - if a response is not required enters the Idle state.
  - If a response is required, it waits for the response.
- When the response is received,
  - for a response with CRC, the received CRC code and the internally generated code are compared, and the appropriate status flag is set according the result.
  - for a response without CRC, no CRC is checked, and the appropriate status flag is not set.

When ever the CPSM is active (not in the Idle state), the CPSMACT bit is set.

Figure 257. Command path state machine (CPSM)



- **Idle**: The command path is inactive. When the command control register is written and the enable bit (CPSMEN) is set, the CPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit) and moves
  - to the Send state when WAITPEND = 0 and BOOTEN = 0.
  - to the Pending state when WAITPEND = 1.
  - to the Boot state when BOOTEN = 1.
- **Send**: The command is sent and the CRC is appended.
  - When CMDTRANS bit is set or when BOOTEN bit is set and BOOTMODE is alternative boot, and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM at the end of the command.
  - When the CMDTRANS bit is set and the CMDSUSPEND bit is 0 the interrupt period is terminated at the end of the command.
  - When CMDSTOP bit is set the CPSM Abort signal is issued to the DPSM at the end of the command.
  - If no response is expected (WAITRESP = 00) the CPSM moves to the Idle state and the CMDSENT flag is set. When BOOTMODE = 1 and BOOTEN = 0 the CMDSENT flag is delayed 56 cycles after the command end bit, otherwise the

CMDSENT flag is generated immediately after the command end bit.
The RESPCMDR and RESPxR registers are not modified.

– If a command response is expected (WAITRESP = not 00) the CPSM moves to the Wait state and start the response timeout.

- **Wait**: The command path waits for a response.
  – When WAITINT bit is 0 the command timer starts running and the CPSM waits for a start bit.
  a) If a start bit is detected before the timeout the CPSM moves to the Receive state.
  b) If the timeout is reached before the CPSM detect a response start bit, the timeout flag (CTIMEOUT) is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are not modified.
  – When WAITINT bit is 1, the timer is disabled and the CPSM waits for an interrupt request (response start bit) from one of the cards.
  a) When a start bit is detected the CPSM moves to the Receive state.
  b) When writing WAITINT to 0 (interrupt mode abort), the host sends a response by its self and on detecting the start bit the CPSM move to the Receive state.
- **Receive**: The command response is received. Depending the response mode bits WAITRESP in the command control register, the response can be either short or long, with CRC or without CRC. The received CRC code when present is verified against the internally generated CRC code.
  – When the CMDSUSPEND bit is set and the SDIO Response bit BS = 0 (response bit [39]), the interrupt period is started after the response.
  When the CMDSUSPEND bit is cleared, or the CMDSUSPEND bit is 1 and the SDIO Response bit BS = 1 (response bit [39]), there is no interrupt period started.
  – When the CMDTRANS bit is set and the CMDSUSPEND bit is set and the SDIO Response bit DF= 1 (response bit [32]) the interrupt period is terminated after the response.
  – When the CRC status passes or no CRC is present the CMDREND flag is set, the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.
  - When BOOTMODE = 1 and BOOTEN = 0 the CMDREND flag is delayed 56 cycles after the response end bit, otherwise the CMDREND flag is generated immediately after the response end bit.
  - When CMDTRANS bit is set and the DTDIR = transmit, the CPSM DataEnable signal is issued to the DPSM at the end of the command response.
  – When the CRC status fails the CCRCFAIL flag is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.
- **Pending**: According the pending WAITPEND bit in the command register, the CPSM enters the pending state.
  – When DATALENGTH ≤ 5 bytes the CPSM moves to the Sent state and generates the DataEnable signal to start the data transfer aligned with the CMD12 Stop Transmission command.
  – When DATALENGTH > 5 bytes, the CPSM DataEnable signal is issued to the DPSM to start the data transfer. The CPSM waits for a send CMD signal from the

DPSM before moving to the Send state. This enables, for example, the CMD12 Stop Transmission command to be sent aligned with the data.

– When writing WAITPEND to 0, the CPSM moves to the Send state.

- **Boot**: If the BOOTEN bit is set in the command register, the CPSM enters the Boot state, and when:
  – BOOTMODE = 0 the SDMMC_CMD line is driven low and when CMDTRANS bit is set and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM. This enables normal boot operation. This state is left at the end of the boot procedure by clearing the register bit BOOTEN, which cause the SDMMC_CMD line to be driven high and the CPSM Abort signal is issued to the DPSM, before moving to the Idle state. The CMDSENT flag is generated 56 cycles after SDMMC_CMD line is high.
  – BOOTMODE = 1, move to the Send state. This enables sending of the CMD0 (boot). Clearing BOOTEN has no effect.

Note: The CPSM remains in the Idle state for at least eight SDMMC_CK periods to meet the N_{CC} and N_{RC} timing constraints. N_{CC} is the minimum delay between two host commands, and N_{RC} is the minimum delay between the host command and the card response.

The response timeout has a fixed value of 64 SDMMC_CK clock periods.

A command is a token that starts an operation. Commands are sent from the host to either a single card (addressed command) or all connected cards (broadcast command are available for *e*•MMC V3.31 or previous). Commands are transferred serially on the SDMMC_CMD line. All commands have a fixed length of 48 bits. The general format for a command token for SD-Memory cards, SDIO cards, and *e*•MMC cards is shown in *Table 213.*

The command token data is taken from two registers, one containing a 32-bit argument and the other containing the 6-bit command index (six bits sent to a card).

Table 213. Command token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	1	Transmission bit
[45:40]	6	x	Command index
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

Next to the command data there are command type (WAITRESP) bits controlling the command path state machine (CPSM). These bits also determine whether the command requires a response, and whether the response is short (48 bit) or long (136 bits) long, and if a CRC is present or not.

A response is a token that is sent from an addressed card or synchronously from all connected cards to the host as an answer to a previous received command. All responses are sent via the command line SDMMC_CMD. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The code length depends on the response type. Response tokens R1, R2, R3, R4, R5, and R6 have various

coding schemes, depending on their content. The general formats for the response tokens for SD-Memory cards, SDIO cards, and *e*•MMC cards are shown in *Table 214*, *Table 215* and *Table 216*.

A response always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (card = 0). A value denoted by x in the tables below indicates a variable entry. Most responses, except some, are protected by a CRC. Every command code word is terminated by the end bit (always 1).

The response token data is stored in five registers, four containing the 32-bits card status, OCR register, argument or 127-bits CID or CSD register including internal CRC, and one register containing the 6-bits command index.

Table 214. Short response with CRC token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

Table 215. Short response without CRC token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	1111111	(reserved 1111111)
0	1	1	End bit

Table 216. Long response with CRC token format

Bit position	Width	Value	Description
135	1	0	Start bit
134	1	0	Transmission bit
[133:128]	6	111111	Reserved
[127:1]	127:8	x	CID or CSD slices
	7:1	x	CRC7 (included in CID or CSD)
0	1	1	End bit

The command/response path operates in a half-duplex mode, so that either commands can be sent or responses can be received. If the CPSM is not in the Send state, the

SDMMC_CMD output is in the Hi-Z state. Data sent on SDMMC_CMD are synchronous with the SDMMC_CK according the NEGEDGE register bit see *Figure 254*.

The command and short response with CRC, the CRC generator calculates the CRC checksum for all 40 bits before the CRC code. This includes the start bit, transmission bit, command index, and command argument (or card status).

For the long response the CRC checksum is calculated only over the 120 bits of R2 CID or CSD. Note that the start bit, transmission bit and the six reserved bits are not used in the CRC calculation.

The CRC checksum is a 7-bit value:

CRC[6:0] = remainder [(M(x) * x^7) / G(x)]

G(x) = $x^7 + x^3 + 1$

M(x) = (first bit) * x^n + (second bit) * x^{n-1}+... + (last bit before CRC) * x^0

Where n = 39 or 119.

The CPSM can send a number of specific commands to handle various operating modes when CPSMEN is set, see *Table 217*.

Table 217. Specific Commands overview

VSWITCH	BOOTEN	BOOTMOD	CMDTRAN	WAITPEND	CMDSTOP	WAITINT	Description
1	x	x	x	x	x	x	Start voltage switch sequence
0	1	x	x	x	x	x	Start normal boot
0	1	1	x	x	x	x	Start alternative boot
0	0	1	x	x	x	x	Stop alternative boot.
0	0	0	1	x	x	x	Send command with associated data transfer.
0	0	0	0	1	1	x	*e*•MMC stream data transfer, command (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	1	0	x	*e*•MMC stream data transfer, command different from (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	0	1	x	Send command (STOP_TRANSMISSION), stopping any ongoing data transmission.
0	0	0	0	0	0	1	Enter *e*•MMC wait interrupt (Wait-IRQ) mode.
0	0	0	0	0	0	0	Any other none specific command

The command/response path implements the status flags and associated clear bits shown in *Table 218*:

Table 218. Command path status flags

Flag	Description
CMDSENT	Set at the end of the command without response (CPSM moves from Send to Idle).
CMDREND	Set at the end of the command response when the CRC is OK (CPSM moves from Receive to Idle).
CCRCFAIL	Set at the end of the command response when the CRC is FAIL (CPSM moves from Receive to Idle).
CTIMEOUT	Set after the command when no response start bit received before the timeout (CPSM moves from Wait to Idle).
CKSTOP	Set after the voltage switch (VSWITCHEN = 1) command response when the CRC is OK and the SDMMC_CK is stopped (no impact on CPSM).
VSWEND	Set after the voltage switch (VSWITCH = 1) timeout of 5 ms + 1 ms (no impact on CPSM).
CPSMACT	Command transfer in progress (CPSM not in Idle state).

The command path error handling is shown in *Table 219*:

Table 219. Command path error handling

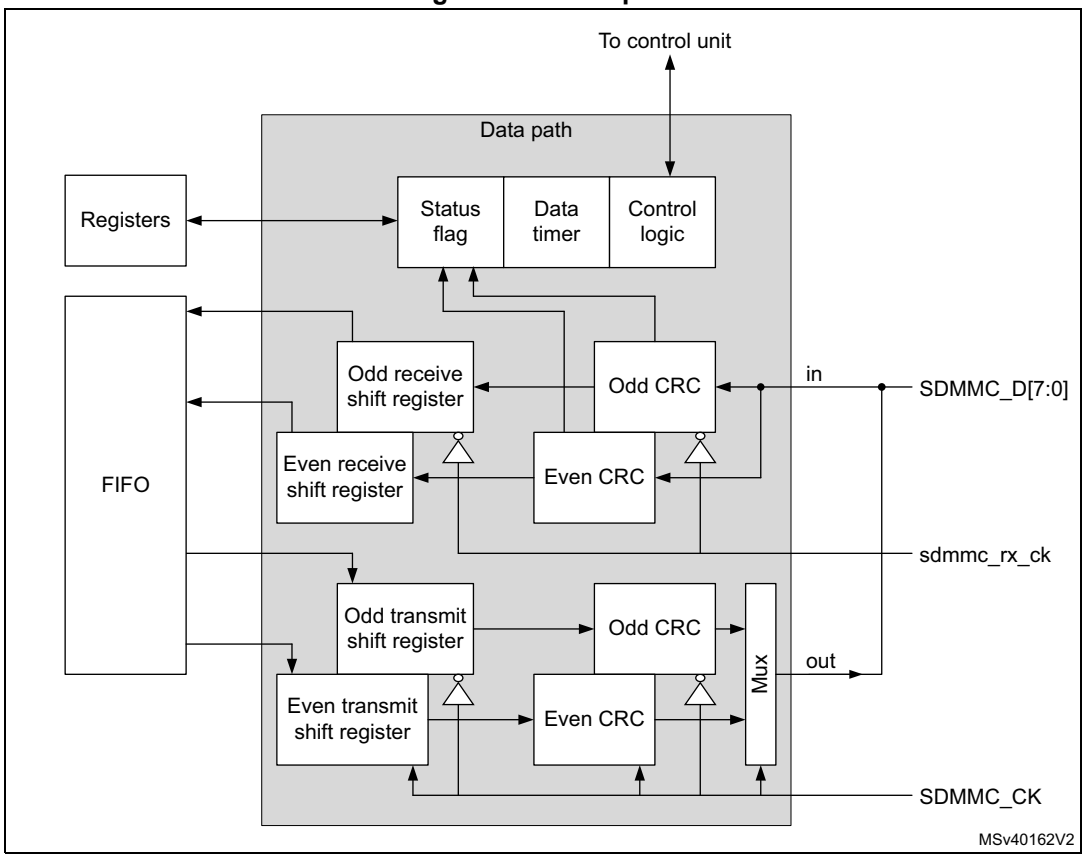
Error	CPSM state	Cause	Card action	Host action	CPSM action
Timeout	Wait	No start bit in time	Unknown	Reset or cycle power card(1)	Move to Idle
CRC status	Receive	Negative status	Command ignored	Resend command(1)	Move to Idle
		Transmission error	Command accepted	Resend command(1)	

1. When CMDTRANS is set, also a stop_transmission command must be send to move the DPSM to Idle.

Data path

The data path subunit transfers data on the SDMMC_D[7:0] lines to and from cards. The data transmit path is clocked on the SDMMC_CK and sends data to the card. The data receive path is clocked on the sdmmc_rx_ck and receives data from the card. *Figure 258* shows the data path block diagram.

Figure 258. Data path



The card data bus width can be programmed in the clock control register bits WIDBUS. The supported data bus width modes are:

- If the wide bus mode is not enabled, only one bit is transferred over SDMMC_D0.
- If the 4-bit wide bus mode is enabled, data is transferred at four bits over SDMMC_D[3:0].
- If the 8-bit wide bus mode is enabled, data is transferred at eight bits over SDMMC_D[7:0].

Next to the data bus width the data sampling mode can be programmed in the clock control register bit DDR. The supported data sampling modes are:

- Single data rate signaling (SDR), data is clocked on the rising edge of the clock.
- Double data rate signaling (DDR), data is clocked on the both edges of the clock. DDR mode is only supported in wide bus mode (4-bit wide and 8-bit wide).

Note: The data sampling mode only applies to the SDMMC_D[7:0] lines. (not applicable to the SDMMC_CMD line.)

In DDR mode, data is sampled on both edges of the SDMMC_CK according the following rules, see also *Figure 259* and *Figure 260*:

- On the rising edge of the clock odd bytes are sampled.
- On the falling edge of the clock even bytes are sampled.
- Data payload size is always a multiple of 2 bytes.
- Two CRC16 are computed per data line
  - Odd bits CRC16 clocked on the falling edge of the clock.
  - Even bits CRC16 clocked on the rising edge of the clock.
- Start, end bits and idle conditions are full cycle.
- CRC status / boot acknowledgment and busy signaling are full cycle and are only sampled on the rising edge of the clock.

In DDR mode, the SDMMC_CK clock division must be ≥ 2.

Figure 259. DDR mode data packet clocking

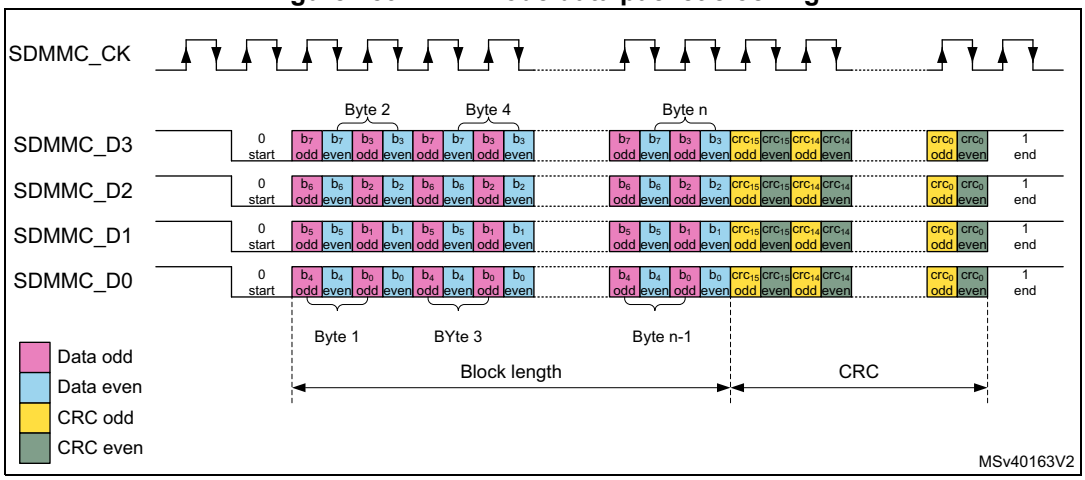
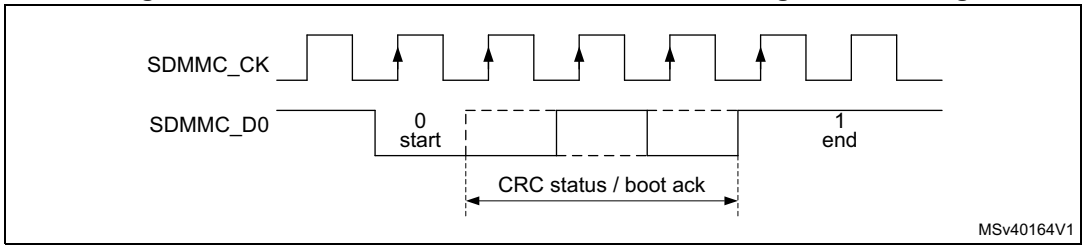


Figure 260. DDR mode CRC status / boot acknowledgment clocking



Data path state machine (DPSM)

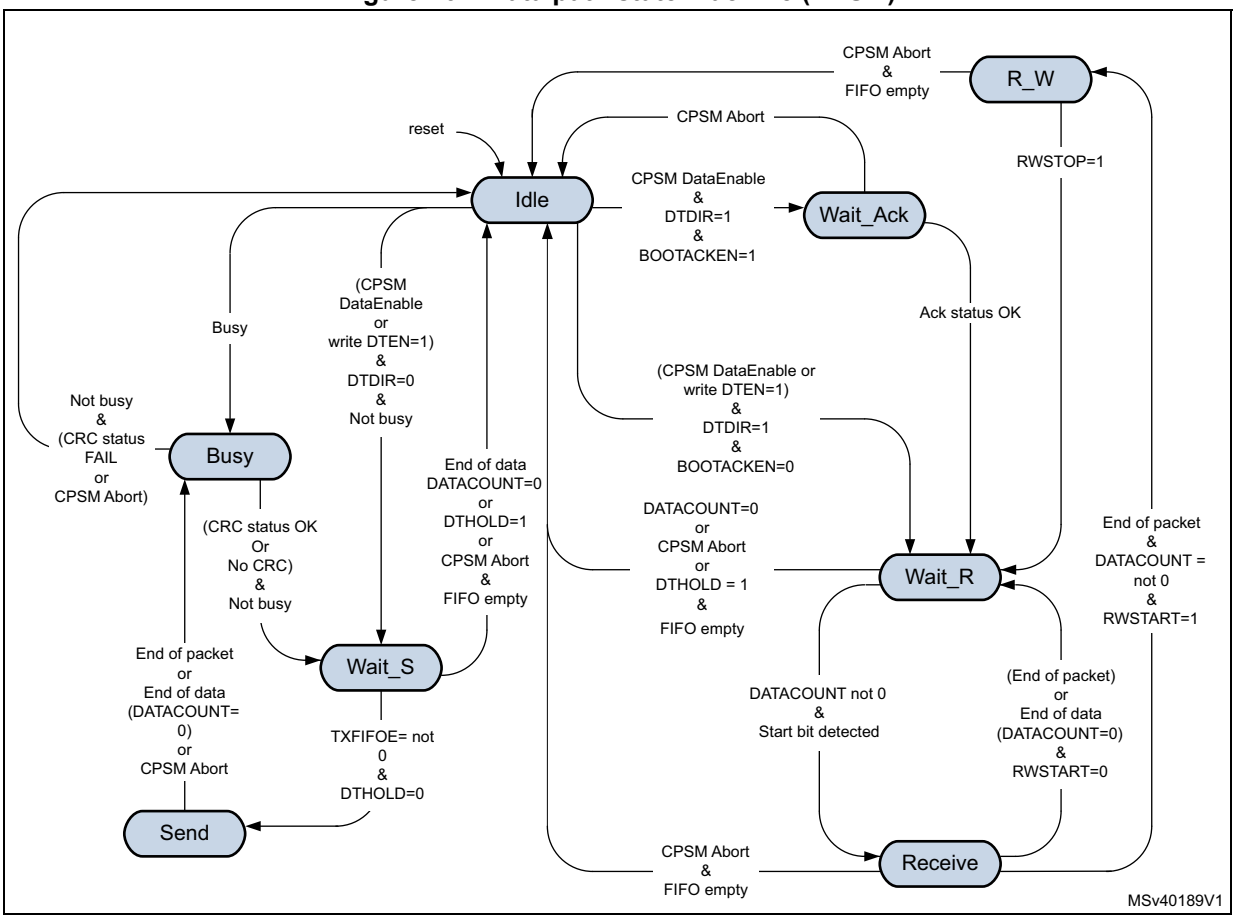
Depending on the transfer direction (send or receive), the data path state machine (DPSM) moves to the Wait_S or Wait_R state when it is enabled:

- Send: the DPSM moves to the Wait_S state. If there is data in the transmit FIFO, the DPSM moves to the Send state, and the data path subunit starts sending data to a card.
- Receive: the DPSM moves to the Wait_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the Receive state, and the data path subunit starts receiving data from a card.

For boot operation with acknowledgment the DPSM moves to the Wait_Ack state and waits for the boot acknowledgment before moving to the Wait_R state.

The DPSM operates at SDMMC_CK. The DPSM has the following states, as shown in *Figure 261*. When ever the DPSM is active (not in the Idle state), the DPSMACT bit is set.

Figure 261. Data path state machine (DPSM)



- **Idle** state: the data path is inactive, and the SDMMC_D[7:0] outputs are according the PWRCTRL setting. The DPSM is activated either by sending a command with CMDTRANS bit set or by setting the DTEN bit, or by detecting Busy on SDMMC_D0 (that is, after a command with R1b response).

  When not busy, the DPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit), loads the data counter with a new (DATALENGTH) value and:

  – When the data direction bit (DTDIR) indicates send, moves to the Wait_S.

  – When the data direction bit (DTDIR) indicates receive, moves to the
  - Wait_R when BOOTACKEN register bit is clear.
  - Wait_Ack when BOOTACKEN register bit is set and start the acknowledgment timeout.

  When busy the DPSM keeps the SDMMC_CK clock active and move to the Busy state.

Note: DTEN must not be used to start data transfer with SD, SDIO and e•MMC cards.

- **Wait_Ack** state: the data path waits for the boot acknowledgment token.
  - The DPSM moves to the Wait_R state if it receives an error free acknowledgment before a timeout.
  - When a pattern different from the acknowledgment is received an acknowledgment status error is generated, and the ack fail status flag (ACKFAIL) is set. The DPSM stays in Wait_Ack.
  - If it reaches a timeout (ACKTIME) before it detects a start bit, it sets the timeout status flag (ACKTIMEOUT). The DPSM stays in Wait_Ack.
  - When the CPSM Abort signal is set it moves to the Idle state and sets the DABORT flag.
- **Wait_R** state: the data path, if the data counter is not zero and data is not hold, waits for a start bit on SDMMC_D[n:0]. If the data counter is zero or data is hold, wait for the FIFO to be empty.
  - In block mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DBLOCKSIZE.
  - In SDIO multibyte mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DATALENGTH.
  - In stream mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data counter with DATALENGTH.
  - if the data counter (DATACOUNT) equals zero (end of data) the DPSM moves to the Idle state when the receive FIFO is empty and the DATAEND flag is set.
  - If it reaches a timeout (DATATIME) before it detects a start bit, it sets the timeout status flag (DTIMEOUT) and the DPSM stays in the Wait_R state.
  - If the CPSM Abort signal is set:
    If DATACOUNT > 0, the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST, and sets the DABORT flag.
    If DATACOUNT is zero normal operation is continued, there is no DABORT flag since the transfer has completed normally.
  - if the DTHOLD bit is set:
    - When DATACOUNT > 0, the DPSM moves to the Idle state when the receive FIFO is empty and when IDMAEN = 0 reset with FIFORST, and issues the DHOLD flag. When holding the timeout is disabled. When an CPSM Abort signal is received during holding, the transfer is aborted.
    - When DATACOUNT = 0, the transfer is completed normally and there is no DHOLD flag.
  - When DPSM has been started with DTEN, after an error (DTIMEOUT) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.

- **R_W** state: the data path Read Wait the bus.
  - The DPSM moves to the Wait_R state when the Read Wait stop bit (RWSTOP) is set, and start the receive timeout.
  - If the CPSM Abort signal is set, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then moves to the Idle state and sets the DABORT flag.
- **Receive** state: the data path receives serial data from a card. Pack the data in bytes and written it to the data FIFO. Depending on the transfer mode selected in the data control register (DTMODE), the data transfer mode can be either block or stream:
  - In block mode, when the data block size (DBLOCKSIZE) number of data bytes are received, the DPSM waits until it receives the CRC code.
  - In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are received, the DPSM waits until it receives the CRC code.

  a) If the received CRC code matches the internally generated CRC code, the DPSM moves to the
  - R_W state when RWSTART = 1 and DATACOUNT > zero, the DBCKEND flag is set.
  - Wait_R state otherwise.

  b) If the received CRC code fails the internally generated CRC code any further data reception is prevented.
  - When not all data has been received (DATACOUNT > 0), the CRC fail status flag (DCRCFAIL) is set and the DPSM stays in the Receive state.
  - When all data has been received (DATACOUNT = 0), wait for the FIFO to be empty after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.

  - In stream mode, the DPSM receives data while the data counter DATACOUNT > 0. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DPSM moves to the Wait_R state.
  - When a FIFO overrun error occurs, the DPSM sets the FIFO overrun error flag (RXOVERR) and any further data reception is prevented. The DPSM stays in the Receive state.
  - When an CPSM Abort signal is received:
  - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted. The remaining data in the shift register is written to the data FIFO, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and the DABORT flag is set.
  - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT = 0, the transfer is completed normally. The DPSM stays in the Receive state no DABORT flag is generated.
  - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or RXOVERR) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.

- **Wait_S** state: the data path waits for data to be available from the FIFO.
  - If the data counter DATACOUNT > 0, waits until the data FIFO empty flag (TXFIFOE) is de-asserted and DTHOLD is not set, and moves to the Send state.
  - If the data counter (DATACOUNT) = 0 the DPSM moves to the Idle state.
    - When DTHOLD is disabled, the DATAEND flag is set.
    - When DTHOLD is enabled, the DHOLD flag is set.
  - When DTHOLD is set and the DATACOUNT > 0
    - When IDMA is enabled, the DBCKEND flag is set and subsequently the FIFO is flushed, furthermore the DPSM moves to the Idle state and the DHOLD flag is set.
    - When IDMA is disabled the DBCKEND flag is set. Wait for the FIFO to be reset by software with FIFORST, then DPSM moves to the Idle state and issues the DHOLD flag.
  - When DTHOLD is set and DATACOUNT = 0 the transfer is completed normally.
  - When receiving the CPSM Abort signal
    - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and sets the DABORT flag.
    - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT = 0, normal operation is continued, there is no DABORT flag since the transfer has completed normally.

Note: The DPSM remains in the Wait_S state for at least two clock periods to meet the N_{WR} timing requirements, where N_{WR} is the number of clock cycles between the reception of the card response and the start of the data transfer from the host.

- **Send** state: the DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block, SDIO multibyte or stream:
  - In block mode, when the data block size (DBLOCKSIZE) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  - In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  - In stream mode, the DPSM sends data to a card while the data counter DATACOUNT > 0. When the data counter reaches zero moves to the Busy state and start the transmit timeout.
    Before sending the last stream byte according to DATACOUNT, the DPSM issues a trigger on the send CMD signal. This signal is used by the CPSM to sent any pending command (CMD12 Stop Transmission command).
  - If a FIFO underrun error occurs, the DPSM sets the FIFO underrun error flag (TXUNDERR). The DPSM stays in the Send state.
  - When receiving the CPSM Abort signal
    - If the CPSM Abort signal is received before the 2 last bits of the transfer with DATACOUNT = 0, the transfer is aborted. The DPSM sends a last data bit followed by an end bit. The FIFO is disabled/flushed, and the DPSM moves to the Busy state to wait for not busy before setting the DABORT flag.
    - If the CPSM Abort signal is received during or after the 2 last bits of the transfer

with DATACOUNT = 0, the transfer is completed normally, there is no DABORT flag.

- **Busy** state: the DPSM waits for the CRC status token when expected, and wait for a not busy signal:
  - If a CRC status token is expected and indicate “non-erroneous transmission” or when there is no CRC expected:
    - it moves to the Wait_S state when SDMMC_D0 is not low (the card is not busy).
    - When the card is busy SDMMC_D0 is low it remains in the Busy state.
  - If a CRC status token is expected and indicates “erroneous transmission”.
    - When not all data has been send (DATACOUNT > 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set. The FIFO is disabled/flushed and the DPSM stays in the Busy state.
    - When all data has been send (DATACOUNT = 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  - If a CRC status (Ncrc) timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - If a busy timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - When receiving the CPSM Abort signal in the Busy state:
    - If the CPSM Abort signal is received before the 2 last bits of the CRC response with DATACOUNT > 0, the data transfer is aborted. The DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
    - If the CPSM Abort signal is received during or after the 2 last bits of the CRC response when DATACOUNT = 0 or when no CRC is expected and DATACOUNT = 0 and there has been no DTIMEOUT error, the DPSM stays in the Busy state no DABORT flag is generated, since the transfer may completed normally.
    - If the CPSM Abort signal is received when a DTIMEOUT error has occurred the DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
  - When entering the Busy state due to an abort in the Send state, the DPSM waits for not busy before moving to the Idle state and the DABORT flag is set.
  - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or DTIMEOUT) the DPSM moves to the Idle state when the FIFO is reset.
  - When the DPSM has been started due to Busy on SDMMC_D0, waits for not busy after which the Busy end status flag (BUSYD0END) is set and the DPSM moves to the Idle state.

The data timer (DATATIME) is enabled when the DPSM is in the Wait_R or Busy state 2 cycles after the data block end bit, or data read command end bit, or R1b response, and generates the data timeout error (DTIMEOUT):

- When transmitting data, the timeout occurs
  - when a CRC status is expected and no start bit is received withing 8 SDMMC_CK cycles, the DTIMEOUT flag is set.
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.
- When receiving data, the timeout occurs
  - when there is still data to be received DATACOUNT > 0 and no start bit is received before the programmed timeout period, the DTIMEOUT flag is set.
- After a R1b response, the timeout occurs
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.

When DATATIME = 0:

- In receive the start bit must be present 2 cycles after the data block end bit or data read command end bit.
- In transmit busy is timed out 2 cycles after the CRC token end bit or stream data end bit.
- After a R1b response busy is timed out 2 cycles after the response end bit.

Data can be transferred from the card to the host (transmit, send) or vice versa (receive). Data are transferred via the SDMMC_Dn data lines, they are stored in a FIFO.

Table 220. Data token format

Description	Start bit	Data(1)	CRC16	End bit	DTMODE
Block data	0	(DBLOCKSIZE, DATALENGTH)	Yes	1	00
SDIO multibyte	0	(DATALENGTH)	Yes	1	01
e•MMC stream	0	(DATALENGTH)	No	1	10

1. The total amount of data to transfer is given by DATALENGTH. Where for Block data the amount of data in each block is given by DBLOCKSIZE.

The data token format is selected with register bits DTMODE according.

The data path implements the status flags and associated clear bits shown in *Table 221*:

Table 221. Data path status flags and clear bits

Flag		Description
DATAEND	TX	Set at the end of the complete data transfer when the CRC is OK and busy has finished and both DTHOLD = 0 and DATACOUNT = 0. (DPSM moves from Wait_S to Idle)
	RX	Set at the end of the complete data transfer when the CRC is OK and all data has been read, (DATACOUNT = 0 and FIFO is empty). (DPSM moves from Wait_R to Idle)
	Boot	

Table 221. Data path status flags and clear bits (continued)

Flag		Description
DCRCFAIL	TX	Set at the end of the CRC when FAIL and busy has finished. (DPSM stay in Busy when there is still data to send and wait for CPSM Abort) (DPSM moves from Busy to Idle when all data has been sent) or DPSM has been started with DTEN
	RX	Set at the end of the CRC when FAIL and FIFO is empty. (DPSM stays in Receive when there is still data to be received and wait for CPSM Abort) (DPSM moves from Receive to Idle when all data has been received or DPSM has been started with DTEN)
	Boot	
ACKFAIL	Boot	Set at the end of the boot acknowledgment when fail. (DPSM stays in Wait_Ack and wait for CPSM Abort)
DTIMEOUT	CMD R1b	Set after the command response no end of busy received before the timeout. (DPSM stays in Busy and wait for CPSM Abort)
	TX	Set when no CRC token start bit received within Ncrc, or no end of busy received before the timeout. (DPSM stays in Busy and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle) Note: The DCRCFAIL flag may also be set when CRC failed before the busy timeout.
	RX	Set when no start bit received before the timeout. (DPSM stays in Wait_R and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle)
	Boot	
ACKTIMEOUT	Boot	Set when no start bit received before the timeout. (DPSM stays in Wait_Ack and wait for CPSM Abort)
DBCKEND	TX	When DTHOLD = 1 and IDMAEN = 0: Set at the end of data block transfer when the CRC is OK and busy has finished, when data transfer is not complete (DATACOUNT >0). (DPSM moves from Busy to Wait_S)
	RX	When RWSTART = 1: Set at the end of data block transfer when the CRC is OK, when data transfer is not complete (DATACOUNT > 0). (DPSM moves from Receive to R_W)
	Boot	
DHOLD	TX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and busy has finished. (DPSM moves from Wait_S to Idle)
	RX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and all data has been read (FIFO is empty), when data transfer is not complete (DATACOUNT >0). (DPSM moves from Wait_R to Idle)
DABORT	CMD R1b	When CPSM Abort event has been sent by the CPSM and busy has finished. (DPSM moves from Busy to Idle)
	TX	When CPSM Abort event has been sent by the CPSM before the 2 last bits of the transfer. (DPSM moves from any state to Idle)
	RX	
	Boot	
BUSYD0END	CMD R1b	Set after the command response when end of busy before the timeout. (DPSM moves from Busy to Idle)
DPSMACT		Data transfer in progress. (DPSM not in Idle state)

The data path error handling is shown in *Table 222*:

Table 222. Data path error handling

Error	DPSM state	Cause	Card action	Host action	DPSM action
Timeout	Wait_Ack	No Ack in time	unknown	Card cycle power	Stay in Wait_Ack (reset the SDMMC with the RCC.SDMMCxRST register bit)
	Wait_R	No start bit in time	unknown	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
			unknown	Stop boot procedure	
	Busy	Busy too long (due to data transfer)	unknown	Stop data reception Send stop transmission command	
		Busy too long (due to R1b)	unknown	Send reset command	
CRC	Receive	transmission error	Send further data	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
CRC status	Busy	Negative status	Ignore further data	Stop data transmission Send stop transmission command	On CPSM Abort move to Idle
		transmission error	wait for further data		
Ack status	Wait_Ack	transmission error	Send boot data	Stop boot procedure	On CPSM Abort move to Idle
Overrun	Receive	FIFO full	Send further data	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
Underrun	Send	FIFO empty	Receive further data	Stop data transmission Send stop transmission command	On CPSM Abort move to Idle

Data FIFO

The data FIFO (first-in-first-out) subunit contains the transmit and receive data buffer. A single FIFO is used for either transmit or receive as selected by the DTDIR bit. The FIFO contain a 32-bit wide, 16-word deep data buffer and control logic. Because the data FIFO operates in the AHB clock domain (sdmmc_hclk), all signals from the subunits in the SDMMC clock domain (SDMMC_CK/sdmmc_rx_ck) are resynchronized.

The FIFO can be in one of the following states:

- The transmit FIFO refers to the transmit logic and data buffer when sending data out to the card. (DTDIR = 0)
- The receive FIFO refers to the receive logic and data buffer when receiving data in from the card. (DTDIR = 1)

The end of a correctly completed SDMMC data transfer from the FIFO is indicated by the DATAEND flags driven by the data path subunit. Any incorrect (aborted) SDMMC data transfer from the FIFO is indicated by one of the error flags (DCRCFAIL, DTIMEOUT, DABORT) driven by the data path subunit, or one of the FIFO error flags (TXUNDERR, RXOVERR) driven by the FIFO control.

The data FIFO can be accessed in the following ways, see *Table 223*.

Table 223. Data FIFO access

Data FIFO access	IDMAEN
From firmware via AHB slave interface	0
From IDMA via AHB master interface	1

Transmit FIFO:

Data can be written to the transmit FIFO when the DPSM has been activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface. The transmit FIFO is accessible via sequential addresses. The transmit FIFO contains a data output register that holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, it increments the read pointer and drives new data out. The transmit FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   – For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in transmit mode (DTDIR = 0).
   – Configures the FIFO in transmit mode.
3. Enable the data transfer
   – either by sending a command from the CPSM with the CMDTRANS bit set
   – or by setting DTEN bit
4. When (DPSMACT = 1) write data to the FIFO.
   – The DPSM stays in the Wait_S state until FIFO is full (TXFIFOF = 1), or the number indicated by DATALENGTH.

- The SDMMC keeps sending data as long as FIFO is not empty, hardware flow control during data transfer is used to prevent FIFO underrun.

5. Write data to the FIFO.
   – When the FIFO is handled by software, wait until the FIFO is half empty (TXFIFOHE flag), write data to the FIFO until FIFO is full (TXFIFOF = 1), or last data has been written.
   – When the FIFO is handled by the IDMA, the IDMA transfers the FIFO date.
6. When last data has been written wait for end of data (DATAEND flag)
   – SDMMC has completely sent all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer error or transfer hold when IDMAEN = 0, firmware must stop writing to the FIFO and flush and reset the FIFO with the FIFORST register bit.

The transmit FIFO status flags are listed in *Table 224*.

Table 224. Transmit FIFO status flags

Flag	Description
TXFIFOF	Set to high when all transmit FIFO words contain valid data.
TXFIFOE	Set to high when the transmit FIFO does not contain valid data.
TXFIFOHE	Set to high when half or more transmit FIFO words are empty.
TXUNDERR	Set to high when an underrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

Receive FIFO:

Data can be read from the receive FIFO when the DPSM is activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface.When the data path subunit receives a word of data, it drives the data on the write databus. The write pointer is incremented after the write operation completes. On the read side, the contents of the FIFO word pointed to by the current value of the read pointer is driven onto the read databus. The receive FIFO is accessible via sequential addresses.

The receive FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   – For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in receive mode (DTDIR = 1).
   – Configures the FIFO in receive mode.
3. Enable the DPSM transfer
   – either by sending a command from the CPSM with the CMDTRANS bit set
   – or by setting DTEN bit.
4. When (DPSMACT = 1) the FIFO is ready to receive data.
   – The DPSM writes the received data to the FIFO.
   - The SDMMC keeps receiving data as long as FIFO is not full, hardware flow control during the data transfer is used to prevent FIFO overrun.
5. Read data from the FIFO.
   – When the FIFO is handled by software, wait until the FIFO is half full (RXFIFOHF flag), read data from the FIFO until FIFO is empty (RXFIFOE = 1).
   - When last data has been received, read data from the FIFO until FIFO is empty (DATAEND = 1).
   – When the FIFO is handled by the IDMA, the IDMA transfers the FIFO date.
6. SDMMC has completely received all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer hold when IDMAEN = 0, the firmware must read the remaining data until the FIFO is empty and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

In case of a data transfer error when IDMAEN = 0, the firmware must stop reading the FIFO and flush and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

The receive FIFO status flags are listed in *Table 225*.

Table 225. Receive FIFO status flags

Flag	Description
RXFIFOF	Set to high when all receive FIFO words contain valid data
RXFIFOE	Set to high when the receive FIFO does not contain valid data.
RXFIFOHF	Set to high when half or more receive FIFO words contain valid data.
RXOVERR	Set to high when an overrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

CLKMUX unit

The CLKMUX selects the source for clock sdmmc_rx_ck to be used with the received data and command response. The receive data clock source can be selected by the clock control register bit SELCLKRX, between:

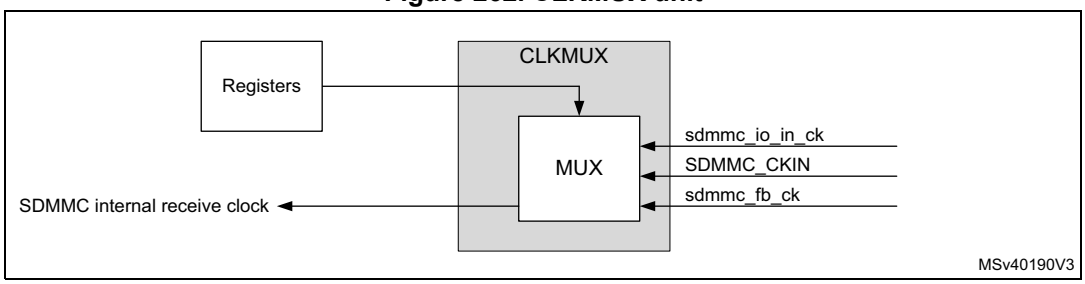
- sdmmc_io_in_ck bus master main feedback clock.
- SDMMC_CKIN external bus feedback clock.
- sdmmc_fb_ck bus tuned feedback clock.

The sdmmc_io_in_ck is selected when there is no external driver, with DS and HS.

The SDMMC_CKIN is selected when there is an external driver with SDR12, SDR25, SDR50 and DDR50.

The sdmmc_fb_ck clock input must be selected when the DLYB block on the device is used with SDR104, HS200 and optionally with SDR50 and DDR50 modes.

Figure 262. CLKMUX unit



The sdmmc_rx_ck source must be changed when the CPSM and DPSM are in the Idle state.

23.5.5 SDMMC AHB slave interface

The AHB slave interface generates the interrupt requests, and accesses the SDMMC adapter registers and the data FIFO. It consists of a data path, register decoder, and interrupt logic.

SDMMC FIFO

The FIFO access is restricted to word access only:

- In transmit FIFO mode
  - Data are written to the FIFO in words (32-bits) until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are written with a word transfer.
- In receive FIFO mode
  - Data are read from the FIFO in words (32-bits) until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are read with a word transfer padded with 0 value bytes.

When accessing the FIFO with half word or byte accesses an AHB bus fault is generated.

SDMMC interrupts

The interrupt logic generates an interrupt request signal that is asserted when at least one of the unmasked status flags is active. A mask register is provided to allow selection of the conditions that generate an interrupt. A status flag generates the interrupt request if a corresponding mask flag is set. Some status flags require an implicit clear in the clear register.

23.5.6 SDMMC AHB master interface

The AHB master interface is used to transfer the data between a memory and the FIFO using the SDMMC IDMA.

SDMMC IDMA

Direct memory access (DMA) is used to provide high-speed transfer between the SDMMC FIFO and the memory. The AHB master optimizes the bandwidth of the system bus. The SDMMC internal DMA (IDMA) provides one channel to be used either for transmit or receive.

The IDMA is enabled by the IDMAEN bit and supports burst transfers of 8 beats.

- In transmit burst transfer mode:
  - Data are fetched in burst from memory whenever the FIFO is empty for the number of burst transfers, until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of the burst size the remaining, smaller then burst size data is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are fetched with a word transfer.
- In receive burst transfer mode:
  - Data are stored in burst in to memory whenever the FIFO contains the number of burst transfers, until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of the burst transfer the remaining, smaller then burst size data, is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are stored with halfword and or byte transfers.

In addition the IDMA provides the following channel configurations selected by bit IDMABMODE:

- single buffered channel
- linked list channel

Single buffered channel

In single buffer configuration the data at the memory side is accessed in a linear matter starting from the base address IDMABASE. When the IDMA has finished transferring all data the and the DPSM has completed the transfer the DATAEND flag is set.

Linked list channel

In linked list configuration, IDMAMODE = 1, the data at the memory side is subsequently accessed from linked buffers, located at base address IDMABASE. The size of the memory buffers is defined by IDMABSIZE. The buffer size must be an integer multiple of the burst size. The bit ULA is used to indicate if a new linked list buffer configuration has to be loaded from the linked list table. A new linked list configuration is loaded when the ULA bit for the current linked list item is set.

The first linked list item configuration is programmed by firmware directly in the SDMMC registers.

When the IDMA has finished transferring all the data of one linked list buffer, according IDMABSIZE, and when the linked list item ULA bit is set, the IDMA loads the new linked list item from the linked list table, and continues transferring data from the next linked list buffer.

When the IDMA has finished transferring all data, according IDMABSIZE and ULA, and the DPSM has completed the transfer, according DATALENGTH, the DATAEND flag is set.

In the following cases, the linked list provides more buffer space than the data to transfer which means the current linked list buffer data has not completely be transfered:

- the ULA bit is set, and all SDMMC data according DATALENGTH has been transfered (DATAEND flag)
- a transfer error (DCRCFAIL when DATACOUNT > 0, RXOVERR, TXUNDERR) occurs
- a transfer is hold (DTHOLD)

In all above cases, the IDMA linked list is stopped and the FIFO is flushed/reset. Before starting or restarting a new SDMMC transfer, the software must initialize a new linked list with correct IDMABASE and IDMABSIZE.

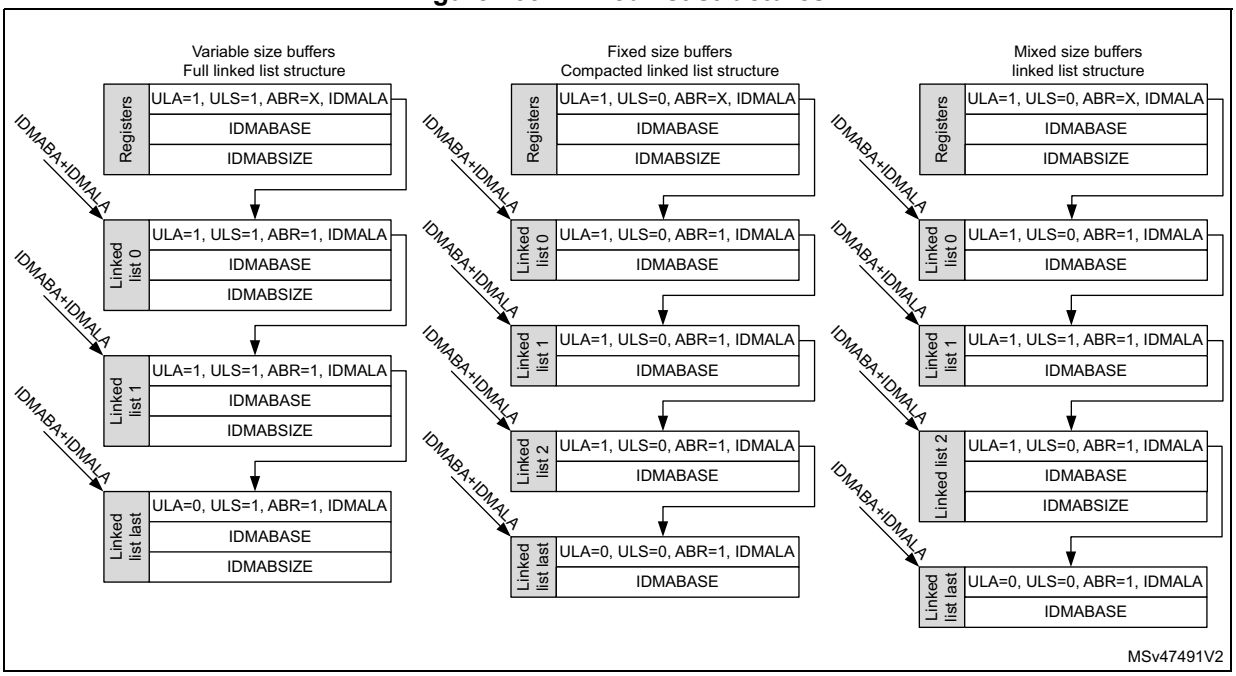
When a IDMA transfer error occurs (see *Section : IDMA transfer error management*) or when the linked list does not provide sufficient buffer space:

- the linked list ends with ULA = 0 and all last linked list buffer data has been transfered, and not all SDMMC data according DATALENGTH has been transfered. The SDMMC transfer is stopped and an IDMA transfer error is generated (see *Section : IDMA transfer error management*).

For a given linked list item, the base address is given by the linked list base IDMABA register value plus the linked list offset IDMALA register value.

The content of each linked list item can be specified by the ULS bit, which makes possible to optionally load the IDMABSIZE, resulting in a 3-word linked list structure. When the IDMABSIZE is not to be loaded (fixed size buffers) a compacted reduced 2-word linked list structure can be used containing only the IDMABASER and the IDMALAR values.

Figure 263. Linked list structures



There is no restriction on mixing both linked list item structures in a single list, this enables the IDMABSIZE to be updated only when needed.

Whenever a linked list buffer has been transfered and the current buffer ULA = 1, an end-of-linked-list-buffer-transfer-complete interrupt (IDMABTC) may be generated (if interrupt is enabled).

Linked list acknowledgment

In the case where software dynamically updates the linked list, during the SDMMC transfer, the availability of a new linked list buffer can be acknowledged by the acknowledge buffer ready (ABR) bit.

When ABR acknowledges that the new linked list buffer is ready, the IDMA continues transferring data from the new linked list buffer.

When ABR indicates that the new linked list buffer is not ready, an IDMA transfer error is generated (see *Section : IDMA transfer error management*). Depending when the IDMA transfer error occurs, it normally causes the generation of an TXUNDERR or RXOVERR error. When a linked list buffer is not acknowledged in time the SDMMC transfer is stopped.

The ABR information is “don't care” when starting the linked list from software programmed register information. The first linked list buffer must be ready to be used before starting the SDMMC transfer.

IDMA transfer error management

An IDMA transfer error can occur:

- When reading or writing a reserved address space (for data or linked list information).
- When there is no more linked list buffer space to store received SDMMC data.
- When all linked list buffer data has been transfered and still more SDMMC data needs to be sent.
- When the availability of a linked list buffer is not acknowledged.

On an IDMA transfer error subsequent IDMA transfers are disabled and an IDMATE flag is set and hardware flow control is disabled. Depending when the IDMA transfer error occurs, it normally causes the generation of a TXUNDERR or RXOVERR error.

The behavior of the IDMATE flag depend on when the IDMA transfer error occurs during the SDMMC transfer:

- An IDMA transfer error is detected before any SDMMC transfer error (TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT):
  - The IDMATE flag is set at the same time as the SDMMC transfer error flag.
  - The TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT interrupt is generated.
- An IDMA transfer error is detected during a STOP_TRANSNMISSION command:
  - The IDMATE flag is set at the same time as the DABORT flag.
  - The DABORT interrupt is generated.
- An IDMA transfer error is detected at the end of the SDMMC transfer (DHOLD, or DATAEND).
  - The IDMATE flag is set at the end of the SDMMC transfer.
  - A SDMMC transfer end interrupt is generated and a DHOLD or DATAEND flag is set.

The IDMATE is generated on an other SDMMC transfer interrupt (TXUNDERR. RXOVERR, DCRCFAIL, DTIMEOUT, DABORT, DHOLD, or DATAEND).

23.5.7 AHB and SDMMC_CK clock relation

The AHB must at least have 3x more bandwidth than the SDMMC bus bandwidth: for example, for SDR50 4-bit mode (50 Mbyte/s), the minimum sdmmc_hclk frequency is 37.5 MHz (150 Mbyte/s).

Table 226. AHB and SDMMC_CK clock frequency relation

SDMMC bus mode	SDMMC bus width	Maximum SDMMC_CK [MHz]	Minimum AHB clock [MHz]
e•MMC DS	8	26	19.5
e•MMC HS	8	52	39
e•MMC DDR52	8	52	78
e•MMC HS200	8	200	150
SD DS / SDR12	4	25	9.4
SD HS / SDR25	4	50	18.8
SD DDR50	4	50	37.5
SD SDR50	4	100	37.5
SD SDR104	4	208	78

23.6 Card functional description

23.6.1 SD I/O mode

The following features are SDMMC specific operations:

- SDIO interrupts
- SDIO suspend/resume operation (write and read suspend)
- SDIO Read Wait operation by stopping the clock
- SDIO Read Wait operation by SDMMC_D2 signaling

Table 227. SDIO special operation control

Operation mode	SDIOEN	RWMOD	RWSTOP	RWSTART	DTDIR
Interrupt detection	1	X	X	X	X
Suspend/Resume operation	X	X	X	X	X
Read Wait SDMMC_CK clock stop (START)	X	1	0	1	1
Read Wait SDMMC_CK clock stop (STOP)	X	1	1	1	1
Read Wait SDMMC_D2 signaling (START)	X	0	0	1	1
Read Wait SDMMC_D2 signaling (STOP)	X	0	1	1	1

SD I/O interrupts

To allow the SD I/O card to interrupt the host, an interrupt function is available on pin 8 (shared with SDMMC_D1 in 4-bit mode) on the SD interface. The use of the interrupt is optional for each card or function within a card. The SD I/O interrupt is level-sensitive, which means that the interrupt line must be held active (low) until it is either recognized and acted upon by the host or deasserted due to the end of the interrupt period. After the host has serviced the interrupt, the interrupt status bit is cleared via an I/O write to the appropriate bit in the SD I/O card internal registers. The interrupt output of all SD I/O cards is active low and the application must provide external pull-up resistors on all data lines (SDMMC_D[3:0]).

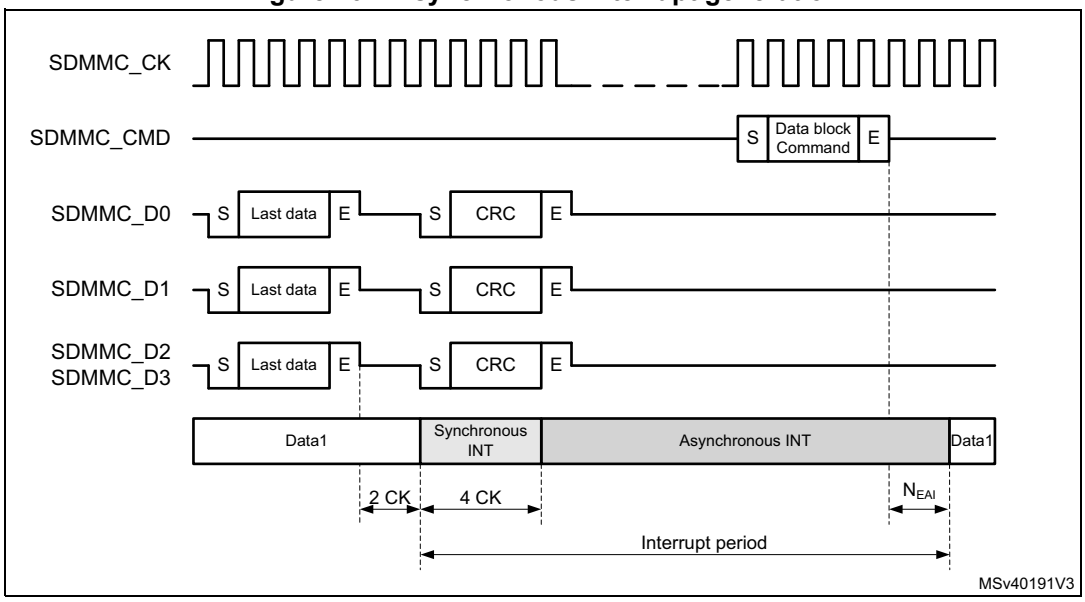
In SD 1-bit mode pin 8 is dedicated to the interrupt function (IRQ), and there are no timing constraints on interrupts.

In SD 4-bit mode the host samples the level of pin 8 (SDMMC_D1/IRQ) into the interrupt detector only during the interrupt period. At all other times, the host interrupt ignores this value. The interrupt period begins when interrupts are enabled at the card and SDIOEN bit is set see register settings in *Table 227*.

In 4-bit mode the card can generate a synchronous or asynchronous interrupt as indicated by the card CCCR register SAI and EAI bits.

- Synchronous interrupt, require the SDMMC_CK to be active.
- Asynchronous interrupt, can be generated when the SDMMC_CK is stopped, 4 cycles after the start of the card interrupt period following the last data block.

Figure 264. Asynchronous interrupt generation



The timing of the interrupt period is depending on the bus speed mode.

In DS, HS, SDR12, and SDR25 mode, selected by register bit BUSSPEED, the interrupt period is synchronous to the SD clock.

- The interrupt period ends at the next clock from the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set.
- The interrupt period resumes 2 SDMMC_CK after the completion of the data block.
- At the data block gap the interrupt period is limited to 2 SDMMC_CK cycles.

Note: DTEN must not be used to start data transfer with SD and e•MMC cards.

Figure 265. Synchronous interrupt period data read

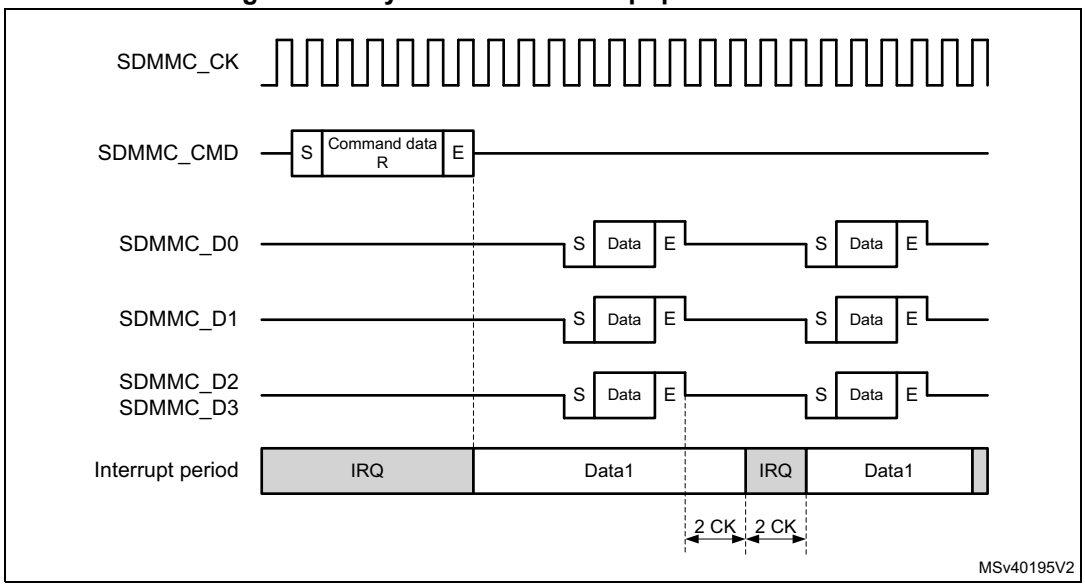
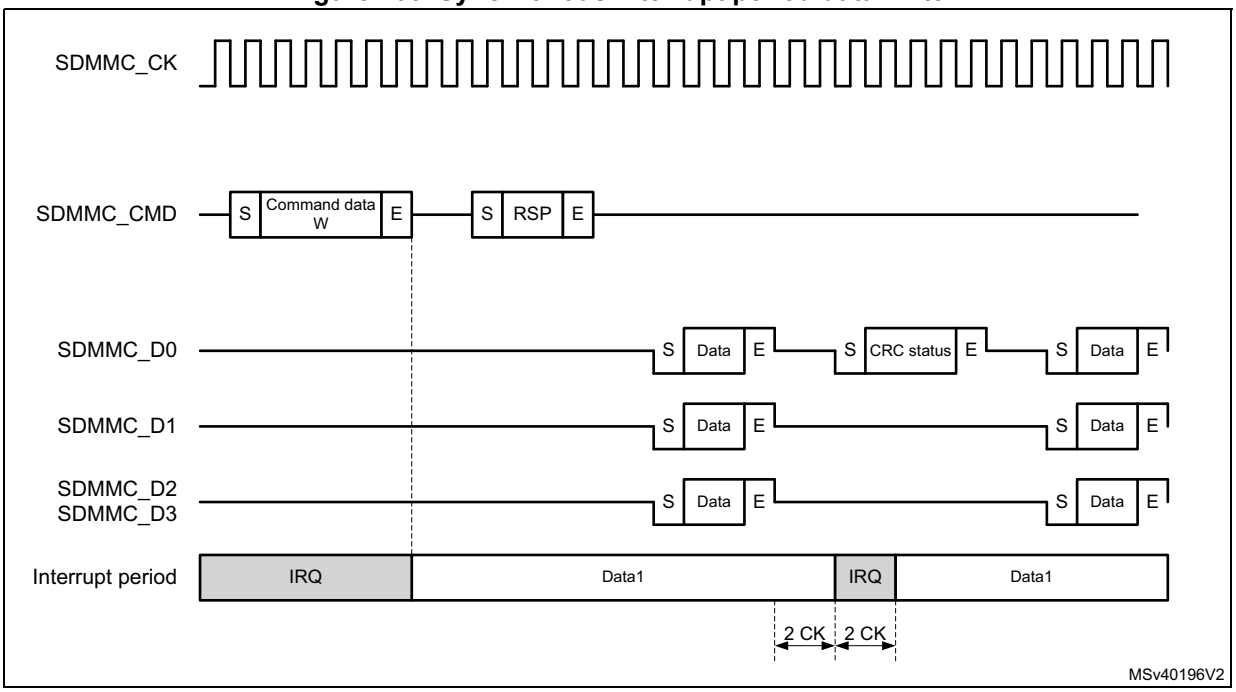


Figure 266. Synchronous interrupt period data write



In SDR50, SDR104, and DDR50, selected by register bit BUSSPEED, due to propagation delay from the card to host, the interrupt period is asynchronous.

- The card interrupt period ends after 0 to 2 SDMMC_CK cycles after the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set. At the host the interrupt period ends after the end bit of a command that transfers data block(s). A card interrupt issued in the 1 to 2 cycles after the command end bit are not detected by the host during this interrupt period.
- The card interrupt period resumes 2 to 4 SDMMC_CK after the completion of the last data block. The host resumes the interrupt period always 2 cycles after the last data block.
- There is NO interrupt period at the data block gap.

Note: DTEN must not be used to start data transfer with SD and e•MMC cards.

Figure 267. Asynchronous interrupt period data read

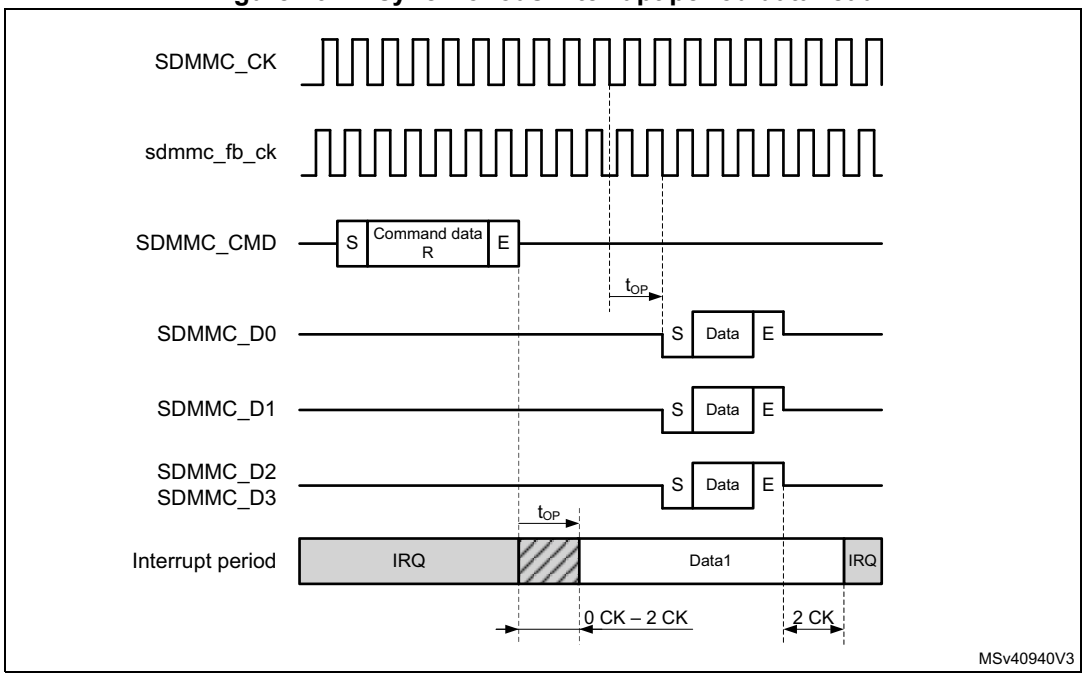
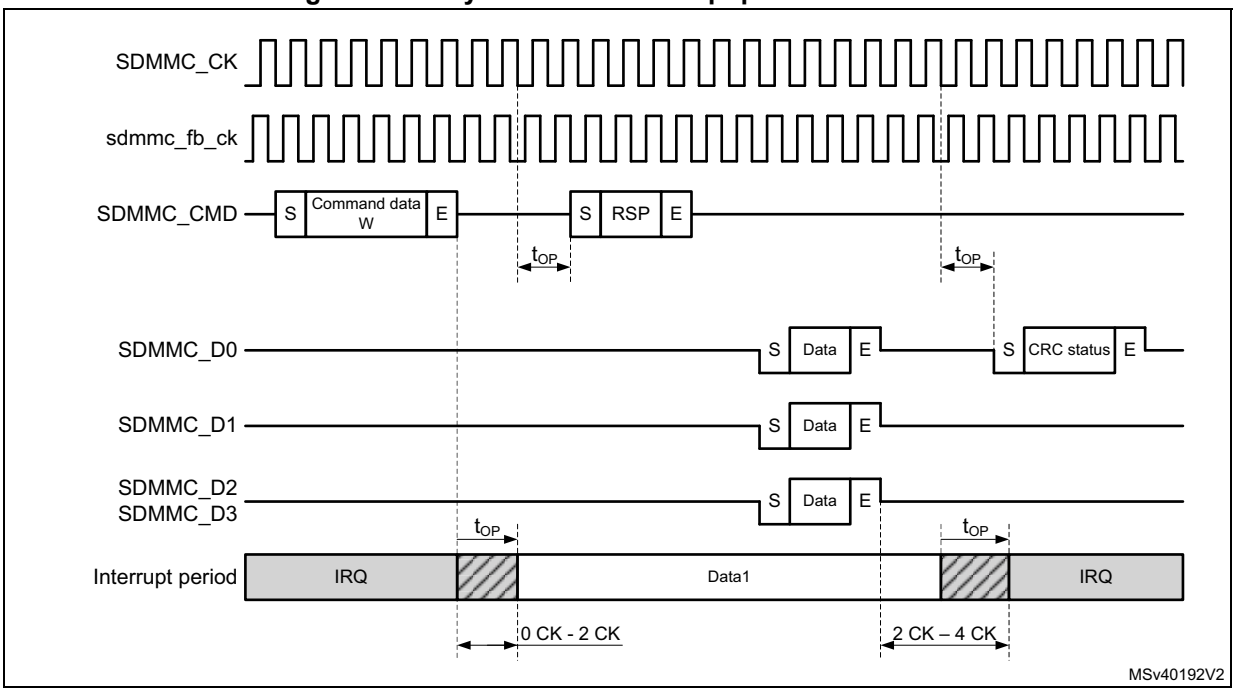


Figure 268. Asynchronous interrupt period data write



When transferring Open-ended multiple block data and using DTMODE “block data transfer ending with STOP_TRANSMISSION command”, the SDMMC masks the interrupt period after the last data block until the end of the CMD12 STOP_TRANSMISSION command.

The interrupt period is applicable for both memory and I/O operations.

In 4-bit mode interrupts can be differentiated from other signaling according *Table 228*.

Table 228. 4-bit mode Start, interrupt, and CRC-status Signaling detection

SDMMC data line	Start	Interrupt	CRC-status
SDMMC_D0	0	1 or CRC-status	0
SDMMC_D1	0	0	X
SDMMC_D2	0	1 or Read Wait	X
SDMMC_D3	0	1	X

SD I/O suspend and resume

This function is NOT supported in SDIO version 4.00 or later.

Within a multifunction SD I/O or a card with both I/O and memory functions, there are multiple devices (I/O and memory) that share access to the *e*•MMC/SD bus. To share access to the host among multiple devices, SD I/O and combo cards optionally implement the concept of suspend/resume. When a card supports suspend/resume, the host can temporarily halt (suspend) a data transfer operation to one function or memory to free the bus for a higher-priority transfer to a different function or memory. After this higher-priority transfer is complete, the original transfer is restarted (resume) where it left off.

To perform the suspend/resume operation on the bus, the host performs the following steps:

1. Determines the function currently using the SDMMC_D[3:0] line(s).
2. Requests the lower-priority or slower transaction to suspend.
3. Waits for the transaction suspension to complete.
4. Begins the higher-priority transaction.
5. Waits for the completion of the higher priority transaction.
6. Restores the suspended transaction.

The card receiving a suspend command responds with its current bus status. Only when the bus has been suspended by the card the bus status indicates suspension completed.

There are different suspend cases conditions:

- Suspend request accepted prior to the start of data transfer.
- Suspend request not accepted, (due to data being transfered at the same time), the host keeps checking the request until it is accepted (data transfer has suspended).
- Suspend request during write busy.
- Suspend request with write multiple.
- Suspend request during Read Wait.

For the host to know if the bus has been released it must check the status of the suspend request, suspension completed.

When the bus status of the suspend request response indicates suspension completed, the card has released the bus. At this time the state of the suspended operation must be saved where after an other operation can start.

The suspend command must be sent with the CMDSUSPEND bit set. This makes possible to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

The hardware does not save the number of remaining data to be transfered when resuming the suspended operation. It is up to firmware to determine the data that has been transferred and resume with the correct remaining number of data bytes.

While receiving data from the card, the SDMMC can suspend the read operation after the read data block end (DPSM in Wait_R). After receiving the suspend acknowledgment response from the card the following steps must be taken by firmware:

1. The normal receive process must be stopped by setting DTHOLD bit.
   a) The remaining number of data bytes in the FIFO must be read until the receive FIFO is empty (RXFIFOE flag is set), and when IDMAEN = 0 the FIFO must be reset with FIFORST.
2. The confirmation that all data has been read from the FIFO, and that the suspend is completed is indicated by the DHOLD flag.
   a) The remaining number of data bytes (multiple of data blocks) still to be read when resuming the operation must be determined from the remaining number of bytes indicated by the DATACOUNT.

Note: When a DTIMEOUT flag occurs during the suspend procedure, this must be ignored.

To resume receiving data from the card, the following steps must be taken by firmware:

1. The remaining number of data bytes (multiple of data blocks) must be programmed in DATALENGTH.
2. The DPSM must be configured to receive data in the DTDIR bit.
3. The resume command must be sent from the CPSM, with the CMDTRANS bit set and the CMDSUSPEND bit set, which ends the interrupt period when data transfer is resumed (response bit DF = 1) and enabled the DPSM, after which the card resumes sending data.

While sending data to the card, the SDMMC can suspend the write operation after the write data block CRC status end (DPSM in Busy). Before sending the suspend command to the card the following steps must be taken by firmware:

1. Enable DHOLD flag (and DBCKEND flag when IDMAEN = 0)
2. The DPSM must be prevented from start sending a new data block by setting DTHOLD.
3. When IDMAEN = 0: When receiving the DBCKEND flag the data transfer is stopped. Firmware can stop filling the FIFO, after which the FIFO must be reset with FIFORST. Any bytes still in the FIFO need to be rewritten when resuming the operation.
4. When receiving the DHOLD flag the data transfer is stopped. The remaining number of data bytes still to be written when resuming must be determined from the remaining number of bytes indicated by the DATACOUNT.
5. To suspend the card the suspend command must be sent by the CPSM with the CMDSUSPEND bit set. This makes possible to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

To resume sending data to the card, the following steps must be taken by firmware:

1. The remaining number of data bytes must be programmed in DATALENGTH.
2. The DPSM must be configured for transmission with DTDIR set and enabled by having the CPSM send the resume command with the CMDTRANS bit set and the CMDSUSPEND bit set. This ends the interrupt period and start the data transfer. The DPSM either goes to the Wait_S state when SDMMC_D0 does not signal busy, or goes to the Busy state when busy is signaled.
3. When IDMAEN = 1: The IDMA needs to be reprogrammed for the remaining bytes to be transfered.
4. When IDMAEN = 0: Firmware must start filling the FIFO with the remaining data.

SD I/O Read Wait

There are two methods to pause the data transfer during the block gap:

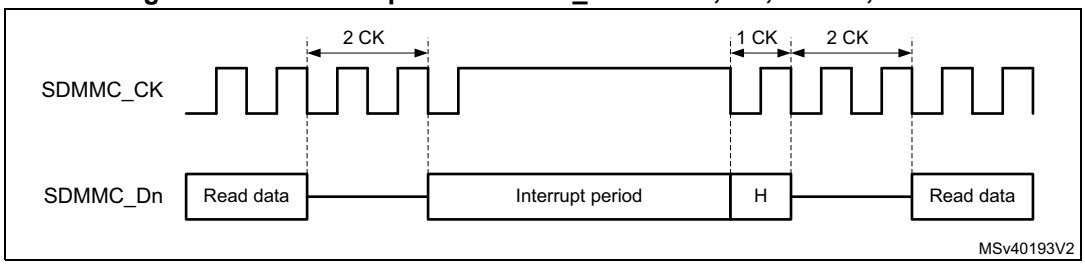
1. Stopping the SDMMC_CK.
2. Using Read Wait signaling on SDMMC_D2.

The SDMMC can perform a Read Wait with register settings according *Table 227*.

Depending the SDMMC operation mode (DS, HS, SDR12, SDR25) or (SDR50, SDR104, DDR) each method has a different characteristic.

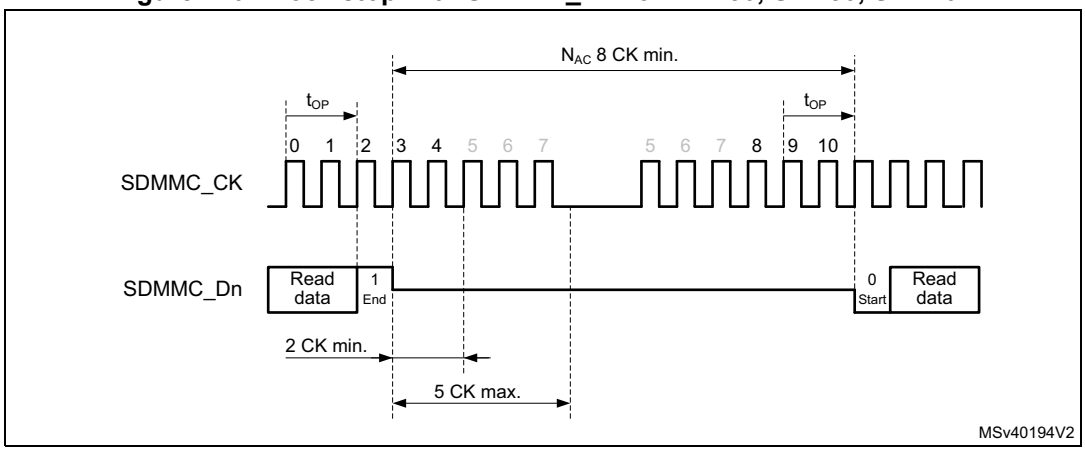
The timing for pause read operation by stopping the SDMMC_CK for DS, HS, SDR12, and SDR25, the SDMMC_CK may be stopped 2 SDMMC_CK cycles after the end bit. When ready the host resumes by restarting clock (see *Figure 269*).

Figure 269. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25



The timing for pause read operation by stopping the SDMMC_CK for SDR50, SDR104, and DDR50, the SDMMC_CK may be stopped minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, after the end bit. When ready the host resumes by restarting clock, see *Figure 270*. (In DDR50 mode the SDMMC_CK must only be stopped after the falling edge, when the clock line is low.)

Figure 270. Clock stop with SDMMC_CK for DDR50, SDR50, SDR104



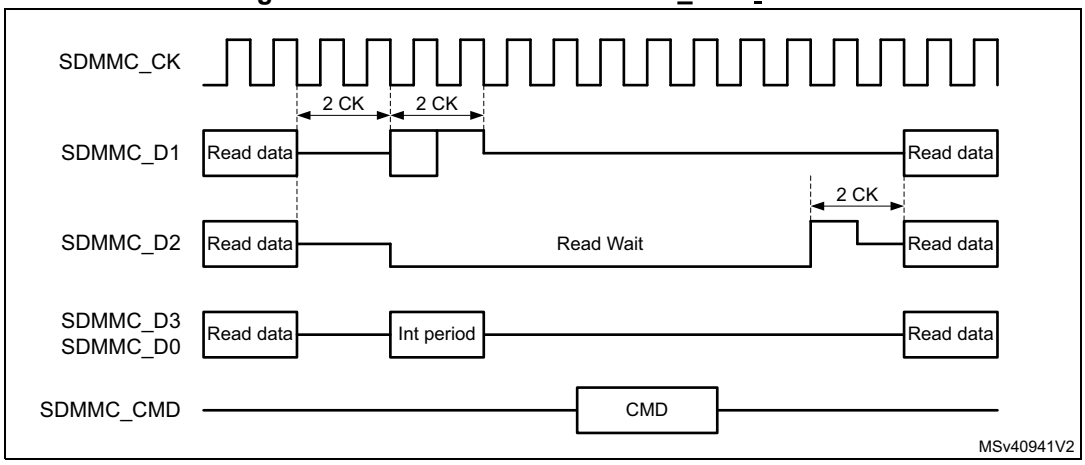
In Read Wait SDMMC_CK clock stopping, when RWSTART is set, the DSPM stops the clock after the end bit of the current received data block CRC. The clock start again after writing 1 to the RWSTOP bit, where after the DPSM waits for a start bit from the card.

As SDMMC_CK is stopped, no command can be issued to the card. During a Read Wait interval, the SDMMC can still detect SDIO interrupts on SDMMC_D1.

The optional Read Wait signaling on SDMMC_D2 (RW) operation is defined only for the SD 1-bit and 4-bit modes. The Read Wait operation enables the host to signal a card that is reading multiple registers (IO_RW_EXTENDED, CMD53) to temporarily stall the data transfer while allowing the host to send commands to any function within the SD I/O device. To determine when a card supports the Read Wait protocol, the host must test capability bits in the internal card registers.

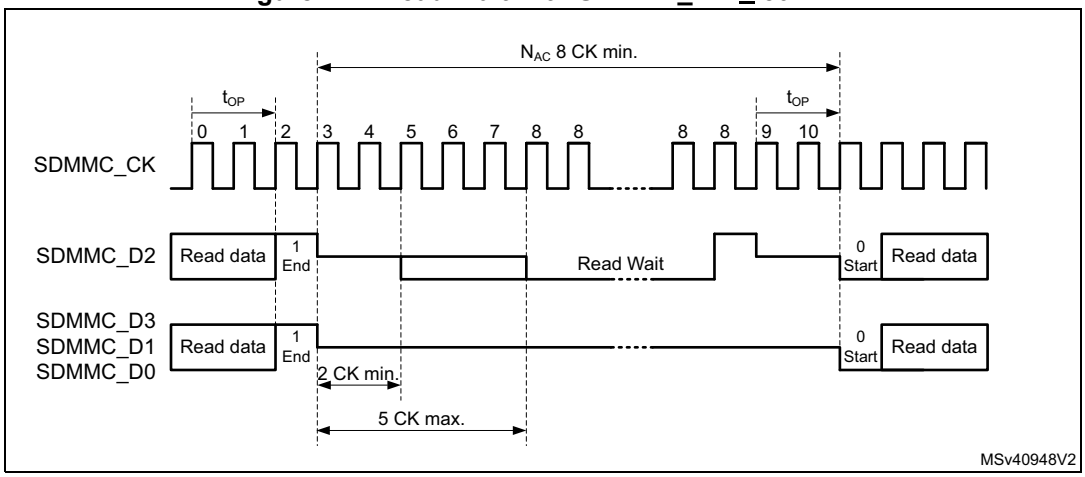
The timing for Read Wait with a SDMMC_CK less then 50MHz (DS, HS, SDR12, SDR25) is based on the interrupt period generated by the card on SDMMC_D1. The host by asserting SDMMC_D2 low during the interrupt period requests the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z, see *Figure 271*.

Figure 271. Read Wait with SDMMC_CK < 50 MHz



For SDR50, SDR104 with a SDMMC_CK more than 50MHz, and DDR50, the card treats the Read Wait request on SDMMC_D2 as an asynchronous event. The host by asserting SDMMC_D2 low after minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, request the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z. The host must raise SDMMC_D2 on the SDMMC_CK clock (see *Figure 272*).

Figure 272. Read Wait with SDMMC_CK ≥ 50 MHz



In Read Wait SDMMC_D2 signaling, when RWSTART is set, the DPSM drives SDMMC_D2 after the end bit of the current received data block CRC. The Read Wait signaling on SDMMC_D2 is removed when writing 1 to the RWSTOP bit. The DPSM remains in R_W state for two more SDMMC_CK clock cycles to drive SDMMC_D2 to 1 for one clock cycle (in accordance with SDIO specification), where after the DPSM waits for a start bit from the card.

During the Read Wait signaling on SDMMC_D2 commands can be issued to the card. During the Read Wait interval, the SDMMC can detect SDIO interrupts on SDMMC_D1.

23.6.2 CMD12 send timing

CMD12 is used to stop/abort the data transfer, the card data transmission is terminated two clock cycles after the end bit of the Stop Transmission command.

Table 229. CMD12 use cases

Data operation	Stop Transmission command CMD12 Description
SDMMC stream write	The data transfer is stopped/aborted by sending the Stop Transmission command.
SDMMC open ended multiple block write	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block write with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block write. (sending the Stop Transmission command after the card has received the last block is regarded as an illegal command.) If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC stream read	The data transfer is stopped/aborted by sending the Stop Transmission command.
SDMMC open ended multiple block read	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block read with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block read. (sending the Stop Transmission command after the card has transmitted the last block is regarded as an illegal command.) Transaction can be aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.

All data write and read commands can be aborted any time by a Stop Transmission command CMD12. The following data abort procedure applies during an ongoing data transfer:

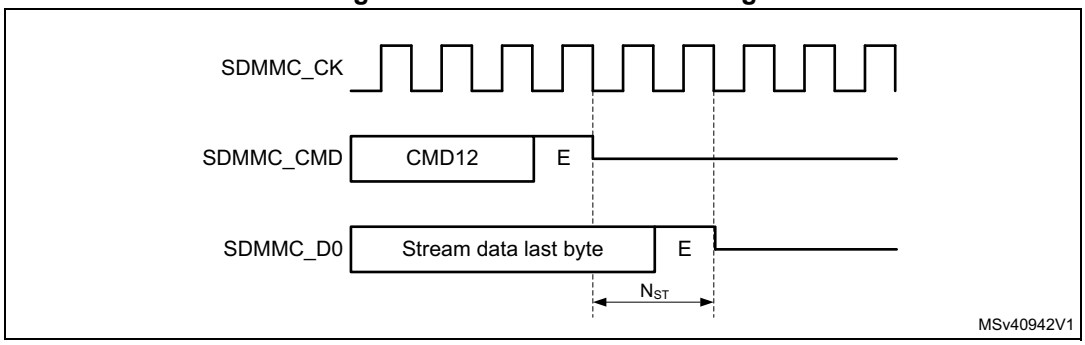
1. Load CMD12 Stop Transmission command in registers and set the CMDSTOP bit.
   – This causes the CPSM Abort signal to be generated when the command is sent to the DPSM.
2. Configure the CPSM to send a command immediately (clear WAITPEND bit).
   – The card, when sending data, stops data transfer 2 cycles after the Stop Transmission command end bit.
   The card when no data is being sent, does not start sending any new data.
   – The host, when sending data, sends one last data bit followed by an end bit after the Stop Transmission command end bit.
   The host when not sending data, does not start sending any new data.
3. When IDMAEN = 0, the FIFO need to be reset with FIFORST.
   – When writing data to the card. On the CMDREND flag, firmware must stop writing data to the FIFO. Subsequently the FIFO must be reset with FIFORST, this flushes the FIFO.
   – When reading data from the card. On the CMDREND flag, firmware must read the remaining data from the FIFO. Subsequently the FIFO must be reset with FIFORST.
4. When IDMAEN = 1, hardware takes care of the FIFO.
   – When writing data to the card. On the CPSM Abort signal, hardware stops the IDMA and subsequently the FIFO is flushed.
   – When reading data from the card. On the CPSM Abort signal, hardware instructs the IDMA to transfer the remaining data from the FIFO to RAM.
5. When the FIFO is empty/reset the DABORT flag is generated.

Stream operation and CMD12

To stop the stream transfer after the last byte to be transfered, the CMD12 end bit timing must be sent aligned with the data stream end of last byte. The following write stream data procedure applies:

1. Initialize the stream data in the DPSM, DTMODE = MCC stream data transfer.
2. Send the WRITE_DATA_STREAM command from the CPSM with CMDTRANS = 1.
3. Preload CMD12 in command registers, with the CMDSTOP bit set.
4. Configure the CPSM to send a command only after a wait pending (WAITPEND = 1) end of last data (according DATALENGTH).
5. Enabling the CPSM to send the STOP_TRANSMISSION command, the stream data end bit and command end bit are aligned.
   – When DATALENGTH > 5 bytes, Command CMD12 is waited in the CPSM to be aligned with the data transfer end bit.
   – When DATALENGTH < 5 bytes, Command CMD12 is started before and the DPSM remains in the Wait_S state to align the data transfer end with the CMD12 end bit.
6. The write stream data can be aborted any time by clearing the WAITPEND bit. This causes the Preloaded CMD12 to be sent immediately and stop the write data stream.

Figure 273. CMD12 stream timing



To stop the read stream transfer after the last byte, the CMD12 end bit timing must occur after the last data stream byte. The following read stream data procedure applies:

1. Wait for all data to be received by the DPSM and read from the FIFO (DATAEND flag).
   – The DPSM does not receive more data than indicated by DATALENGTH, even if the card is sending more data.
2. Send CMD12 by the CPSM.
   – CMD12 stops the card sending data.

Note: The SDMMC does not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

Block operation and CMD12

To stop block transfer at the end of the data, the CMD12 end bit must be sent after the last block end bit.

When writing data to the card the CMD12 end bit must be sent after the write data block CRC token end bit. This requires the CMD12 sending to be tied to the data block transmission timing. To stop an Open-ended Multiple block write, the following procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be sent by the DPSM and the CRC token to be received, (DATAEND flag).
   – The DPSM does not send more data than indicated by DATALENGTH.
3. Send CMD12 by the CPSM.
   – CMD12 sets the card to Idle mode.

When reading data from the card the CMD12 end bit must be sent earliest at the same time as the card read data block last data bit. This requires the CMD12 sending to be tied to the data block reception timing. The following stop Open-ended Multiple block read data block procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be received by the DPSM and read from the FIFO (DATAEND flag).
   – The DPSM does not receive more data than indicated by DATALENGTH, even if the card is sending more data.
3. Send CMD12 with CMDSTOP bit set by the CPSM.
   – CMD12 stops the Card sending more data and set the card to Idle mode. Any ongoing block transfer is aborted by the Card.

Note: The SDMMC does not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

23.6.3 Sleep (CMD5)

The *e*•MMC card may be switched between a Sleep state and a Standby state by CMD5. In the Sleep state the power consumption of the card is minimized and the Vcc power supply may be switched off.

The CMD5 (SLEEP) is used to initiate the state transition from Standby state to Sleep state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Sleep state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Enable interrupt on BUSYD0END.
2. Send CMD5 (SLEEP).
3. On BUSYD0END interrupt, card is in Sleep state.
4. Vcc power supply can be switched off.

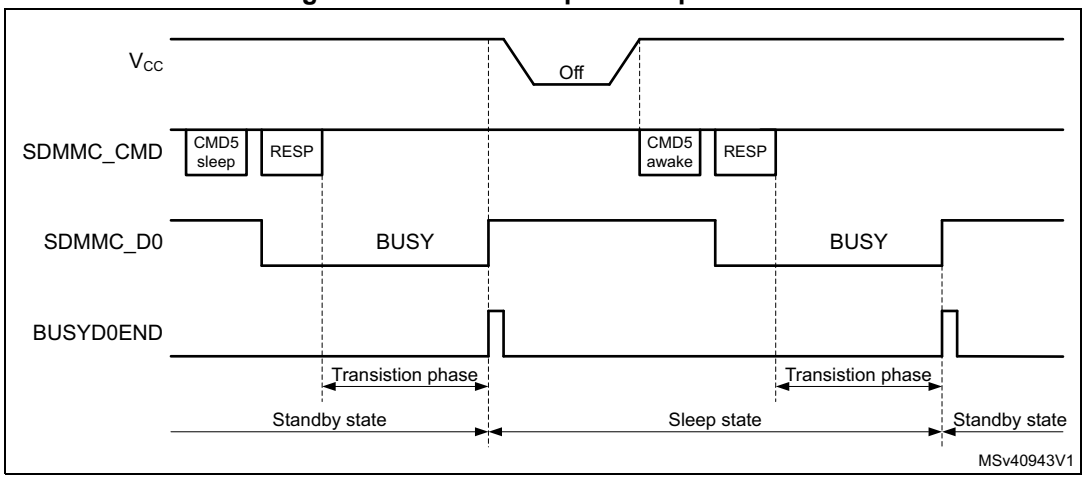
The CMD5 (AWAKE) is used to initiate the state transition from Sleep state to Standby state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Standby state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Switch on Vcc power supply and wait unit minimum operating level is reached.
2. Enable interrupt on BUSYD0END.
3. Send CMD5 (AWAKE).
4. On BUSYD0END interrupt card is in Standby state.

The Vcc power supply can be switched off only after the Sleep state has been reached. The Vcc supply must be reinstalled before CMD5 (AWAKE) is sent.

Figure 274. CMD5 Sleep Awake procedure



23.6.4 Interrupt mode (Wait-IRQ)

The host and card enter and exit interrupt mode (Wait-IRQ) simultaneously. In interrupt mode there is no data transfer. The only message allowed is an interrupt service request response from the card or the host. For the interrupt mode to work correctly the SDMMC_CK frequency must be set in accordance with the achievable SDMMC_CMD data rate in Open Drain mode, which depend on the capacitive load and pull-up resistor. The CLKDIV must be set >1, and the SETCLKRX must select either the sdmmc_io_in_ck or SDMMC_CLKin source.

The host must ensure that the card is in Standby state before issuing the CMD40 (GO_IRQ_STATE). While waiting for an interrupt response the SDMMC_CK clock signal must be kept active.

A card in interrupt mode (IRQ state):

- is waiting for an internal card interrupt event. Once the event occurs, the card starts to send the interrupt service request response. The response is sent in open-drain mode.
- while waiting for the internal card interrupt event, the card also monitors the SDMMC_CMD line for a start bit. Upon detection of a start bit the card aborts the interrupt mode and switch to Standby state.

The host in interrupt mode (CPSM Wait state waiting for interrupt):

- is waiting for a card interrupt service request response (start bit).
- while waiting for a card interrupt service request response the host may abort the interrupt mode (by clearing the WAITINT register bit), which causes the host to send a interrupt service request response R5 with RCA = 0x0000 in open-drain mode.

When sending the interrupt service request response, the sender bit-wise monitors the SDMMC_CMD bit stream. The sender whose interrupt service request response bit does not correspond to the bit on the SDMMC_CMD line stops sending. In the case of multiple senders only one successfully sends its full interrupt service request response. If the host sends simultaneously, it loses sending after the transmission bit.

To handle the interrupt mode, the following procedure applies:

1. Set the SDMMC_CK frequency in accordance with the achievable SDMMC_CMD data rate in Open-drain mode, CLKDIV must be set >1, and SETCLKRX must select the sdmmc_io_in_ck.
2. Load CMD40 (GO_IRQ_STATE) in the command registers.
3. Enable wait for interrupt by setting WAITINT register bit.
4. Configure the CPSM to send a command immediately.
   – This causes the CMD40 to be sent and the CPSM to be halted in the Wait state, waiting for a interrupt service request response.
5. To exit the wait for interrupt state (CPSM Wait state):
   – Upon the detection of an interrupt service request response start bit the CPSM moves to the Receive state where the response is received. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.
   – To abort the interrupt mode the host clears the WAITINT register bit, which causes the host to send an interrupt service request response by itself. This moves the CPSM to the Receive state. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.

Note: On a simultaneous send interrupt service request response start bit collision the host loses the bus access after the transmission bit.

23.6.5 Boot operation

In boot operation mode the host can read boot data from the card by either one of the two boot operation functions:

- Normal boot. (keeping CMD line low)
- Alternative boot (sending CMD0 with argument 0xFFFFFFFA)

The boot data can be read according the following configuration options, depending on card register settings:

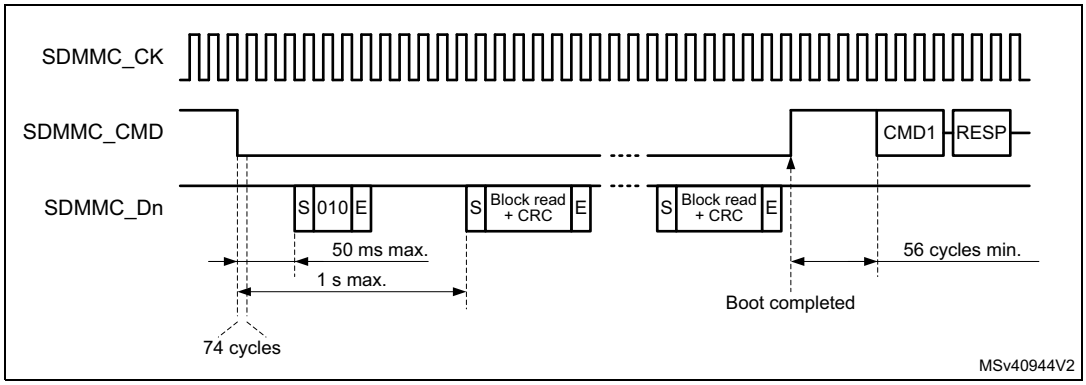
- The partition from which boot data is read (EXT_CSD Byte[179])
- The boot data size (EXT_CSD Byte[226])
- The bus configuration during boot (EXT_CSD Byte[177])
- Receiving boot acknowledgment from the card. (EXT_CSD Byte[179])

If boot acknowledgment is enabled the card send pattern 010 on SDMMC_D0 within 50ms after boot mode has been requested by either CMD line going low or after CMD0 with argument 0xFFFFFFFA. A boot acknowledgment timeout (ACKTIMEOUT) and acknowledgment status (ACKFAIL) is provided.

Normal boot operation

If the SDMMC_CMD line is held low for at least 74 clock cycles after card power-up or reset, before the first command is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD line goes low, the card starts to sent the first boot code data on the SDMMC_Dn line(s). The host must keep the SDMMC_CMD line low until after all boot data has been read. The host can terminate boot mode by pulling the SDMMC_CMD line high.

Figure 275. Normal boot mode operation



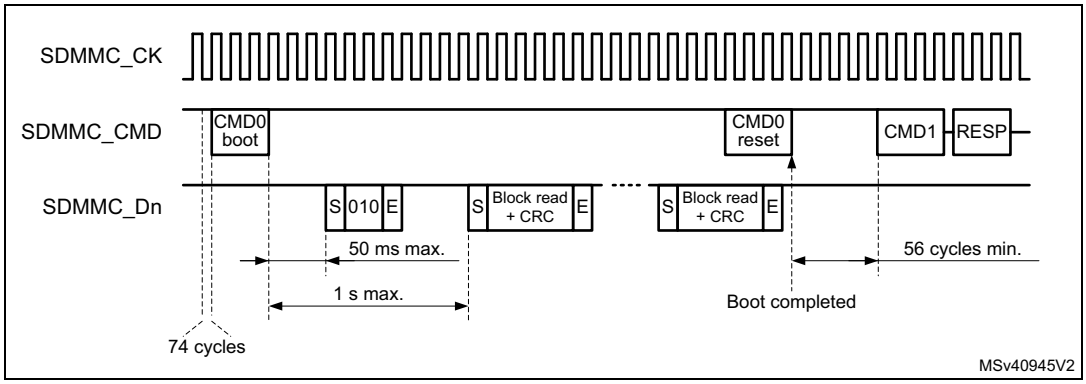
To perform the normal boot procedure the following steps needed:

1. Reset the card.
2. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKFAIL and ACKTIMEOUT interrupt.
3. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data bytes to be received in DATALENGTH.
4. Enable the DTIMEOUT, DATAEND, and CMDSENT interrupts for end of boot command confirmation.
5. Select the normal boot operation mode in BOOTMODE, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN.This causes:
   – the SDMMC_CMD to be driven low. (BOOTMODE = normal boot).
   – the ACK timeout to start.
   – DPSM to be enabled.
6. The incorrect reception of the boot acknowledgment can be detected with ACKFAIL flag or ACKTIMEOUT flag when enabled.
   – when an incorrect boot acknowledgment is received the ACKFAIL flag occurs.
   – when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
7. when all boot data has been received the DATAEND flag occurs.
   – when data CRC fails the DCRCFAIL flag is also generated.
   – when the data timeout occurs the DTIMEOUT flag is also generated.
8. When last data has been received, read data from the FIFO until FIFO is empty after which end of data DATAEND flag is generated.
   – SDMMC has completely received all data and the DPSM is disabled.
9. The boot procedure is terminated by firmware clearing BOOTEN, which causes the SDMMC_CMD line to go high. The CMDSENT flag is generated 56 cycles later to indicate that a new command can be sent.
   – If the boot procedure is aborted by firmware before all data has been received the CPSM Abort signal stops data reception and disables the DPSM which triggers an DABORT flag when enabled.
10. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command.

Alternative boot operation

After card power-up or reset, if the host send CMD0 with the argument 0xFFFFFFFA after 74 clock cycles before CMD0 is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD0 with argument 0xFFFFFFFA has been sent, the card starts to send the first boot code data on the SDMMC_Dn line(s). The master terminates boot operation by sending CMD0 (Reset).

Figure 276. Alternative boot mode operation



To perform the alternative boot procedure the following steps needed:

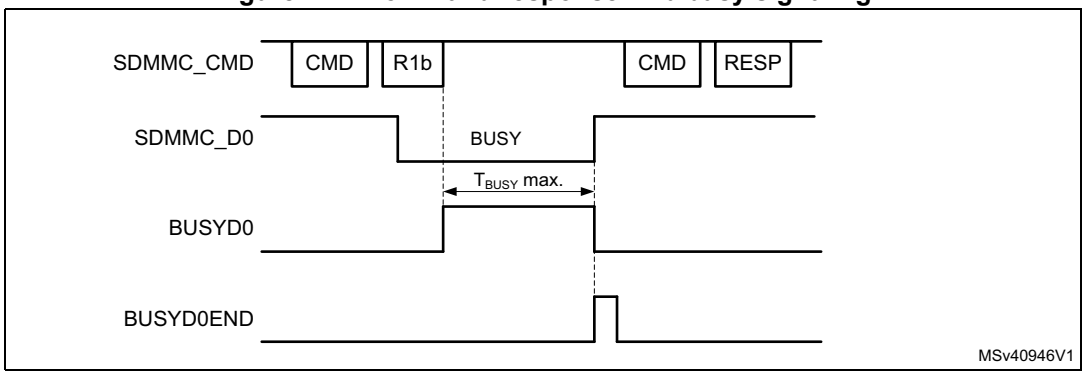
1. Move the SDMMC to power-off state, and reset the card.
2. Move the SDMMC to power-on state. This guarantees the 74 SCDMMC_CK cycles to be clocked before any command.
3. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKTIMEOUT flag.
4. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data to be received in DATALENGTH. Enable the DTIMEOUT and DATAEND flags.
5. Select the alternative boot operation mode in BOOTMODE, load the CMD0 with the 0xFFFFFFFA argument in the command registers. Enable CMDSENT flag for end of boot command confirmation, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN. This causes:
   - the loaded command and argument to be sent out. (BOOTMODE = alternative boot).
   - the ACK timeout to start.
   - DPSM to be enabled.
6. When the command has been sent the CMDSENT flag is generated, at which time the BOOTEN bit must be cleared.
7. the reception of the boot acknowledgment can be detected with ACKFAIL flag when enabled.
   - when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
8. when all boot data has been received the DATAEND flag occurs.
   - when data CRC fails the DCRCFAIL flag is also generated.
   - when the data timeout occurs the DTIMEOUT flag is also generated.

9. When last data has been received, read data from the FIFO until FIFO is empty after which end of data DATAEND flag is generated.
   – SDMMC has completely received all data and the DPSM is disabled.
10. The BOOTEN bit must be cleared, before terminating the boot procedure by sending CMD0 (Reset) with BOOTMODE = alternative boot. This causes the CMDSENT flag to occur 56 cycles after the Command.
   – if the boot procedure is aborted by firmware before all data has been received the CPSM Abort signal stops the data transfer and disable the DPSM which triggers an DABORT flag when enabled.
11. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command. When the RESET command has been sent successfully, the BOOTMODE control bit has to be cleared to terminate the boot operation.

23.6.6 Response R1b handling

When sending commands which have a R1b response the busy signaling is reflected in the BUSYD0 register bit and the release of busy with the BUSYD0END flag. The SDMMC_D0 line is sampled at the end of the R1b response and signaled in the BUSYD0 register bit. The BUSYD0 register bit is reset to not busy when the SDMMC_D0 line release busy, at the same time the BUSYD0END flag is generated.

Figure 277. Command response R1b busy signaling



The expected maximum busy time must be set in the DATATIME register before sending the command. When enabled, the DTIMEOUT flag is set when after the R1b response busy stays active longer then the programmed time.

To detect the SDMMC_D0 busy signaling when sending a Command with R1b response the following procedure applies:

- Enable CMDREND flag.
- Send Command through CPSM.
- On the CMDREND flag check the BUSYD0 register bit.
  – If BUSYD0 signals not busy, signal busy release to firmware
  – If BUSYD0 signals busy, wait for BUSYD0END flag
- On BUSYD0END flag signal busy released to firmware.
- On DTIMEOUT flag busy is active longer then programmed time.

23.6.7 Reset and card cycle power

Reset

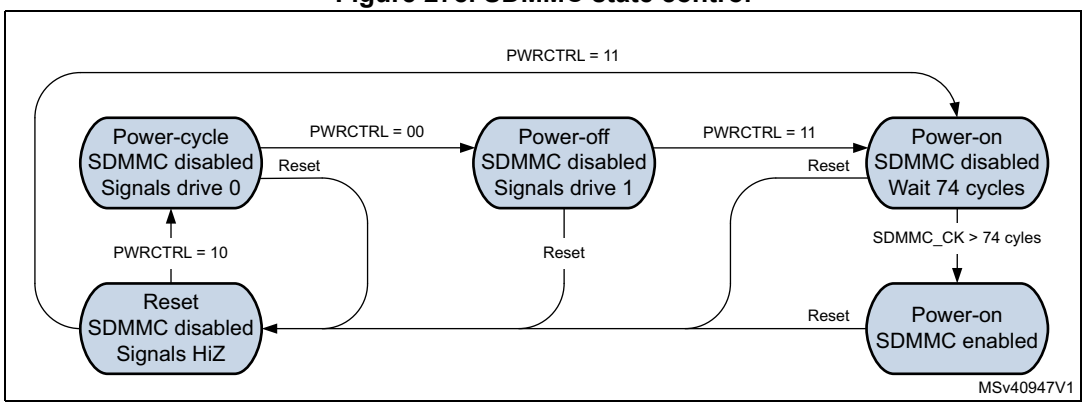
Following reset the SDMMC is in the reset state. In this state the SDMMC is disabled and no command nor data can be transfered. The SDMMC_D[7:0], and SDMMC_CMD are in HiZ and the SDMMC_CK is driven low.

Before moving to the power-on state the SDMMC must be configured.

In the power-on state the SDMMC_CK clock is running. First 74 SDMMC_CK cycles are clocked after which the SDMMC is enabled and command and data can be transfered.

The SDMMC states are controlled by Firmware with the PWRCTL register bits according *Figure 278.*.

Figure 278. SDMMC state control

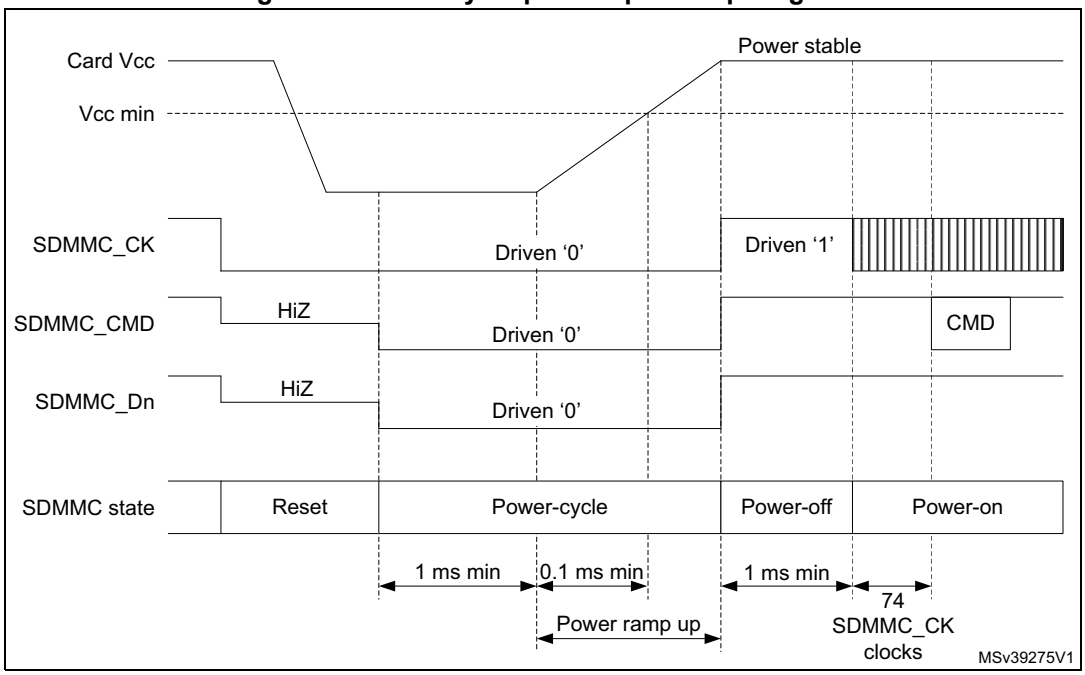


Card cycle power

To perform a card cycle power the following procedure applies:

1. Reset the SDMMC with the RCC.SDMMCxRST register bit. This resets the SDMMC to the reset state and the CPSM and DPSM to the Idle state.
2. Disable the Vcc power to the card.
3. Set the SDMMC in power-cycle state. This makes that the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low, to prevent the card from being supplied through the signal lines.
4. After minimum 1 ms enable the Vcc power to the card.
5. After the power ramp period set the SDMMC to the power-off state for minimum 1 ms. The SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are set to drive 1.
6. After the 1 ms delay set the SDMMC to power-on state in which the SDMMC_CK clock is enabled.
7. After 74 SDMMC_CK cycles the first command can be sent to the card.

Figure 279. Card cycle power / power up diagram



23.7 Hardware flow control

The hardware flow control during data transfer functionality is used to avoid FIFO underrun (TX mode) and overrun (RX mode) errors.

The behavior is to stop SDMMC_CK during data transfer and freeze the SDMMC state machines. The data transfer is stalled when the FIFO is unable to transmit or receive data. The data transfer remains stalled until the transmit FIFO is half full or all data according DATALENGTH has been stored, or until the receive FIFO is half empty. Only state machines clocked by SDMMC_CK are frozen, the AHB interfaces are still alive. The FIFO can thus be filled or emptied even if flow control is activated.

On an IDMA linked list transfer error, the hardware flow control is disabled. As a consequence, depending on when the IDMA linked list transfer error occurs, an underrun or overrun error may also occur (see *Section : IDMA transfer error management*).

To enable hardware flow control during data transfer, the HWFC_EN register bit must be set to 1. After reset hardware flow control is disabled.

Hardware flow control must only be used when the SDMMC_Dn receive data is cycle-aligned with the SDMMC_CK or when sending data. When receiving data and the sdmmc_fb_ck from the DLYB delay block is used. In the case of SDR104 receive mode with a t_{OP} and Dt_{OP} delay > 1 cycle, hardware flow control can not be used.

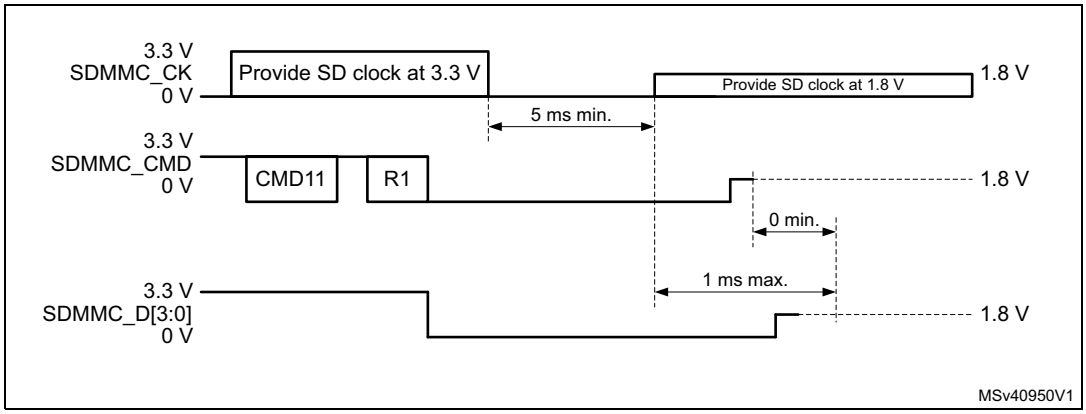
Warning: When variable delay is used and hardware flow control is enabled, sending commands during data transfer may disturb the command response reception. (CMD12 is not

impacted by this, except when CMD12 is used to abort during data transfer.)

23.8 Ultra-high-speed phase I (UHS-I) voltage switch

UHS-I mode (SDR12, SDR25, SDR50, SDR104, and DDR50) requires the support for 1.8 V signaling. After power up the card starts in 3.3V mode. CMD11 invokes the voltage switch sequence to the 1.8V mode. When the voltage sequence is completed successfully the card enters UHS-I mode with default SDR12 and card input and output timings are changed.

Figure 280. CMD11 signal voltage switch sequence



To perform the signal voltage switch sequence the following steps are needed:

1. Before starting the Voltage Switch procedure, the SDMMC_CK frequency must be set in the range 100 kHz - 400 kHz.
2. The host starts the Voltage Switch procedure by setting the VSWITCHEN bit before sending the CMD11.
3. The card returns an R1 response.
   - if the response CRC is pass, the Voltage Switch procedure continues the host does no longer drive the CMD and SDMMC_D[3:0] signals until completion of the voltage switch sequence. Some cycles after the response the SDMMC_CK is stopped and the CKSTOP flag is set.
   - if the response CRC is fail (CCRCFAIL flag) or no response is received before the timeout (CTIMEOUT flag), the Voltage Switch procedure is stopped.
4. The card drives CMD and SDMMC_D[3:0] to low at the next clock after the R1 response.
5. The host, after having received the R1 response, may monitor the SDMMC_D0 line using the BUSYD0 register bit. The SDMMC_D0 line is sampled two SDMMC_CK clock cycles after the Response. The Firmware may read the BUSYD0 register bit following the CKSTOP flag.
   - When the BUSYD0 is detected low the host firmware switches the Voltage regulator to 1.8V, after which it instructs the SDMMC to start the timing critical

section of the Voltage Switch sequence by setting register bit VSWITCH. The hardware continues to stop the SDMMC_CK by holding it low for at least 5 ms.

– When the BUSYD0 is detected high the host aborts the Voltage Switch sequence and cycle power the card.

6. The card after detecting SDMMC_CK low begins switching signaling voltage to 1.8 V.
7. The host SDMMC hardware after at least 5 ms restarts the SDMMC_CK.
8. The card within 1 ms from detecting SDMMC_CK transition drives CMD and DAT[3:0] high for at least 1 SDMMC_CK cycle and then stop driving CMD and DAT[3:0].
9. The host SDMMC hardware, 1 ms after the SDMMC_CK has been restarted, the SDMMC_D0 is sampled into BUSYD0 and the VSWEND flag is set.
10. The host, on the VSWEND flag, checks SDMMC_D0 line using the BUSYD0 register bit, to confirm completion of voltage switch sequence:
    – When BUSYD0 is detected high, Voltage Switch has been completed successfully.
    – When BUSYD0 is detected low, Voltage Switch has failed, the host cycles the card power.

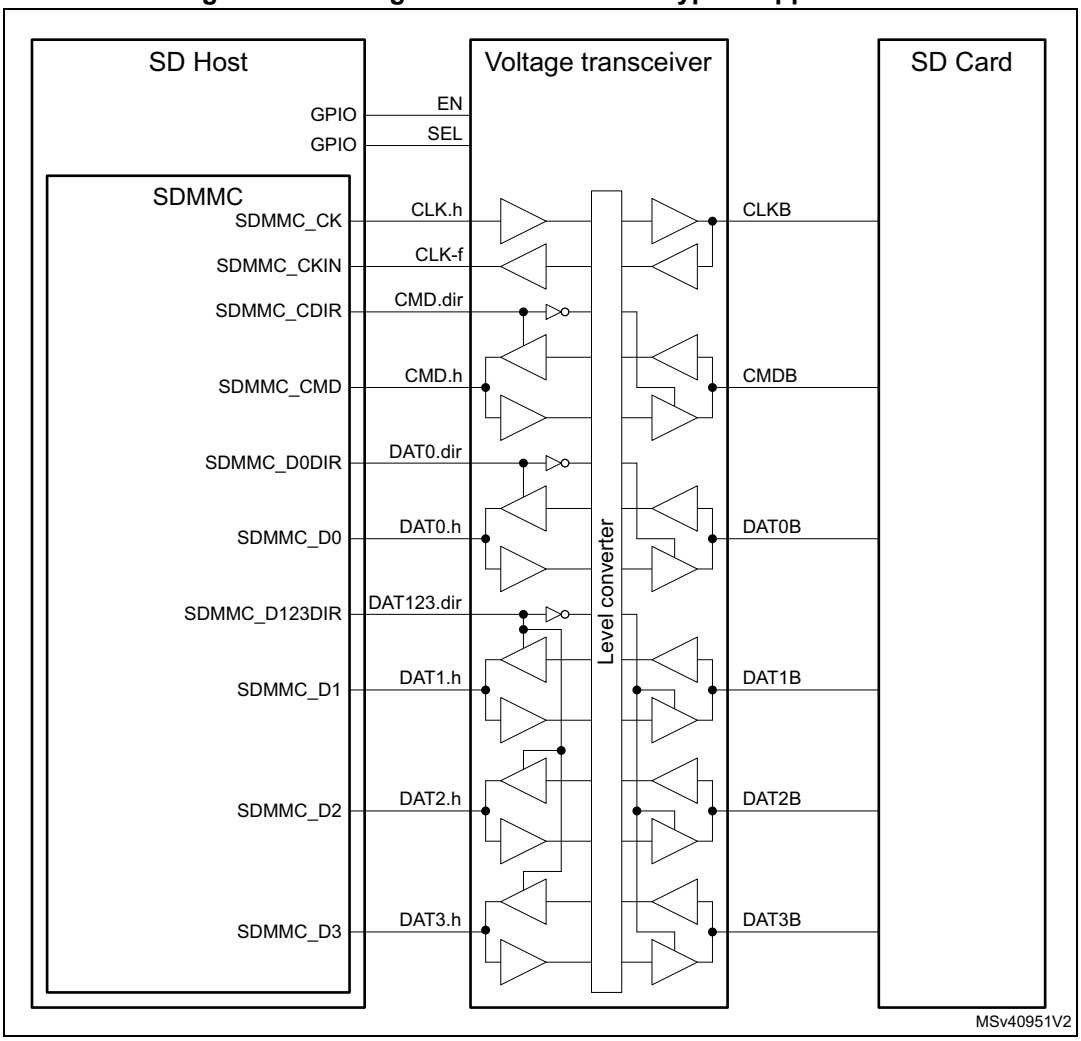
The minimum 5 ms time to stop the SDMMC_CK is derived from the internal un-gated SDMMC_CK clock, which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The >5 ms time is counted by 2^{12} cycles (10.24 ms @ 400 kHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time for the SDMMC_CK clock to be stopped is longer.

The maximum 1 ms time for the card to drive the SDMMC_Dn and SDMMC_CMD lines high is derived from the internal ungated SDMMC_CK which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The SDMMC checks the lines after >1 ms time which is counted by 2^9 cycles (1.28 ms @ 25 MHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time to check the lines is longer.

The signal voltage level switch is supported through an external voltage translation transceiver.

Note: For external devices not needing a signal voltage level switch the signal voltage level can be kept static for all SDMMC operations, including UHS-I.

Figure 281. Voltage switch transceiver typical application



To interface with an external driver (a voltage switch transceiver), next to the standard signals the SDMMC uses the following signals:

- **SDMMC_CKIN** feedback input clock
- **SDMMC_CDIR** I/O direction control for the CMD signal
- **SDMMC_D0DIR** I/O direction control for the SDMMC_D0 signal
- **SDMMC_D123DIR** I/O direction control for the SDMMC_D1, SDMMC_D2 and SDMMC_D3 signals

The voltage transceiver signals **EN** and **SEL** are to be handled through general-purpose I/O.

The polarity of the SDMMC_CDIR, SDMMC_D0DIR and SDMMC_D123DIR signals can be selected through SDMMC_POWER.DIRPOL control bit.

23.9 SDMMC interrupts

Table 230. SDMMC interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode
SDMMC	Command response CRC fail	CCRCFAIL	CCRCFAILIE	CCRCFAILC	Yes
SDMMC	Data block CRC fail	DCRCFAIL	DCRCFAILIE	DCRCFAILC	Yes
SDMMC	Command response timeout	CTIMEOUT	CTIMEOUTIE	CTIMEOUTC	Yes
SDMMC	Data timeout	DTIMEOUT	DTIMEOUTIE	DTIMEOUTC	Yes
SDMMC	Transmit FIFO underrun	TXUNDERR	TXUNDERRIE	TXUNDERRC	Yes
SDMMC	Receive FIFO overrun	RXOVERR	RXOVERRIE	RXOVERRC	Yes
SDMMC	Command reponse received	CMDREND	CMDRENDIE	CMDRENDC	Yes
SDMMC	Comand sent	CMDSENT	CMDSENTIE	CMDSENTC	Yes
SDMMC	Data transfer ended	DATAEND	DATAENDIE	DATAENDC	Yes
SDMMC	Data transfer hold	DHOLD	DHOLDIE	DHOLDC	Yes
SDMMC	Data block sent or received	DBCKEND	DBCKENDIE	DBCKENDC	Yes
SDMMC	Data transfer aborted	DABORT	DABORTIE	DABORTC	Yes
SDMMC	Transmit FIFO half empty	TXFIFOHE	TXFIFOHEIE	N/A	Yes
SDMMC	Receive FIFO half full	RXFIFOHF	RXFIFOHFIE	N/A	Yes
SDMMC	Transmit FIFO full	TXFIFOF	N/A	N/A	Yes
SDMMC	Receive FIFO full	RXFIFOF	RXFIFOFIE	N/A	Yes
SDMMC	Transmit FIFO empty	TXFIFOE	TXFIFOEIE	N/A	Yes
SDMMC	Receive FIFO empty	RXFIFOE	N/A	N/A	Yes
SDMMC	Command response end of busy	BUSYD0END	BUSYD0ENDIE	BUSYD0ENDC	Yes
SDMMC	SDIO interrupt	SDIOIT	SDIOITIE	SDIOITC	Yes
SDMMC	Boot acknowledgment fail	ACKFAIL	ACKFAILIE	ACKFAILC	Yes
SDMMC	Boot acknowledgment timeout	ACKTIMEOUT	ACKTIMEOUTIE	ACKTIMEOUTC	Yes
SDMMC	Voltage switch timing	VSWEND	VSWENDIE	VSWENDC	Yes
SDMMC	SDMM_CK stopped in voltage switch	CKSTOP	CKSTOPIE	CKSTOPC	Yes
SDMMC	IDMA transfer error	IDMATE	IDMATEIE	IDMATEC	Yes
SDMMC	IDMA buffer transfer complete	IDMABTC	IDMABTCIE	IDMABTCC	Yes

23.10 SDMMC registers

The device communicates to the system via 32-bit control registers accessible via AHB slave interface.

The peripheral registers have to be accessed by words (32-bit). Byte (8-bit) and half-word (16-bit) accesses trigger an AHB bus error.

23.10.1 SDMMC power control register (SDMMC_POWER)

Address offset: 0x000

Reset value: 0x0000 0000

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

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

Bit 4 **DIRPOL**: Data and command direction signals polarity selection

This bit can only be written when the SDMMC is in the power-off state (PWRCTRL = 00).

0: Voltage transceiver I/Os driven as output when direction signal is low.

1: Voltage transceiver I/Os driven as output when direction signal is high.

Bit 3 **VSWITCHEN**: Voltage switch procedure enable

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

This bit is used to stop the SDMMC_CK after the voltage switch command response:

0: SDMMC_CK clock kept unchanged after successfully received command response.

1: SDMMC_CK clock stopped after successfully received command response.

Bit 2 **VSWITCH**: Voltage switch sequence start

This bit is used to start the timing critical section of the voltage switch sequence:

0: Voltage switch sequence not started and not active.

1: Voltage switch sequence started or active.

Bits 1:0 **PWRCTRL[1:0]**: SDMMC state control bits

These bits can only be written when the SDMMC is not in the power-on state (PWRCTRL ≠ 11).

These bits are used to define the functional state of the SDMMC signals:

00: After reset, Reset: the SDMMC is disabled and the clock to the Card is stopped, SDMMC_D[7:0], and SDMMC_CMD are HiZ and SDMMC_CK is driven low. When written 00, power-off: the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven high.

01: Reserved (When written 01, PWRCTRL value does not change)

10: Power-cycle, the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low.

11: Power-on: the card is clocked, The first 74 SDMMC_CK cycles the SDMMC is still disabled. After the 74 cycles the SDMMC is enabled and the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are controlled according the SDMMC operation.

Any further write is ignored, PWRCTRL value keeps 11.

23.10.2 SDMMC clock control register (SDMMC_CLKCR)

Address offset: 0x004

Reset value: 0x0000 0000

This register controls the SDMMC_CK output clock, the sdmmc_rx_ck receive clock, and the bus width.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]		BUS SPEED	DDR	HWFC_EN	NEG EDGE
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WID BUS[1:0]		Res.	PWR SAV	Res.	Res.	CLKDIV[9:0]									
rw	rw		rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 21:20 **SELCLKRX[1:0]**: Receive clock selection

These bits can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: sdmmc_io_in_ck selected as receive clock

01: SDMMC_CKIN feedback clock selected as receive clock

10: sdmmc_fb_ck tuned feedback clock selected as receive clock

11: Reserved (select sdmmc_io_in_ck)

Bit 19 **BUSSPEED**: Bus speed for selection of SDMMC operating modes

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: DS, HS, SDR12, SDR25, Legacy compatible, High speed SDR, High speed DDR bus speed mode selected

1: SDR50, DDR50, SDR104, HS200 bus speed mode selected

Bit 18 **DDR**: Data rate signaling selection

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

DDR rate must only be selected with 4-bit or 8-bit wide bus mode. (WIDBUS > 00). DDR = 1 has no effect when WIDBUS = 00 (1-bit wide bus).

DDR rate must only be selected with clock division > 1 (CLKDIV > 0).

0: SDR Single data rate signaling

1: DDR double data rate signaling

Bit 17 **HWFC_EN**: Hardware flow control enable

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: Hardware flow control is disabled

1: Hardware flow control is enabled

When Hardware flow control is enabled, the meaning of the TXFIFOE and RXFIFOF flags change, see SDMMC status register definition in *Section 23.10.11*.

Bit 16 **NEGEDGE**: SDMMC_CK dephasing selection bit for data and command

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

When clock division = 1 (CLKDIV = 0), this bit has no effect. Data and Command change on SDMMC_CK falling edge.

0: When clock division >1 (CLKDIV > 0) and DDR = 0:

- Command and data changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

When clock division >1 (CLKDIV > 0) and DDR = 1:

- Command changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
- Data changed on the sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

1: When clock division >1 (CLKDIV > 0) and DDR = 0:

- Command and data changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.

When clock division >1 (CLKDIV > 0) and DDR = 1:

- Command changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.
- Data changed on the SDMMC_CK falling edge succeeding a SDMMC_CK edge.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

Bits 15:14 **WIDBUS[1:0]**: Wide bus mode enable bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: Default 1-bit wide bus mode: SDMMC_D0 used (Does not support DDR)

01: 4-bit wide bus mode: SDMMC_D[3:0] used

10: 8-bit wide bus mode: SDMMC_D[7:0] used

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PWRSAV**: Power saving configuration bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

For power saving, the SDMMC_CK clock output can be disabled when the bus is idle by setting PWRSAV:

0: SDMMC_CK clock is always enabled

1: SDMMC_CK is only enabled when the bus is active

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **CLKDIV[9:0]**: Clock divide factor

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

This field defines the divide factor between the input clock (sdmmc_ker_ck) and the output clock (SDMMC_CK): SDMMC_CK frequency = sdmmc_ker_ck / [2 * CLKDIV].

0x000: SDMMC_CK frequency = sdmmc_ker_ck / 1 (Does not support DDR)

0x001: SDMMC_CK frequency = sdmmc_ker_ck / 2

0x002: SDMMC_CK frequency = sdmmc_ker_ck / 4

0x0XX: ..

0x080: SDMMC_CK frequency = sdmmc_ker_ck / 256

0xXXX: ..

0x3FF: SDMMC_CK frequency = sdmmc_ker_ck / 2046

Note: While the SD/SDIO card or e•MMC is in identification mode, the SDMMC_CK frequency must be less than 400 kHz.

The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.

At least seven sdmmc_hclk clock periods are needed between two write accesses to this register. SDMMC_CK can also be stopped during the Read Wait interval for SD I/O cards: in this case the SDMMC_CLKCR register does not control SDMMC_CK.

23.10.3 SDMMC argument register (SDMMC_ARGR)

Address offset: 0x008

Reset value: 0x0000 0000

This register contains a 32-bit command argument, which is sent to a card as part of a command message.

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

Bits 31:0 **CMDARG[31:0]**: Command argument

These bits can only be written by firmware when CPSM is disabled (CPSMEN = 0). Command argument sent to a card as part of a command message. If a command contains an argument, it must be loaded into this register before writing a command to the command register.

23.10.4 SDMMC command register (SDMMC_CMDR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register contains the command index and command type bits. The command index is sent to a card as part of a command message. The command type bits control the command path state machine (CPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMD SUS PEND
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOT EN	BOOT MODE	DT HOLD	CPSM EN	WAITP END	WAIT INT	WAITRESP[1:0]		CMD STOP	CMD TRANS	CMDINDEX[5:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **CMDSUSPEND**: The CPSM treats the command as a Suspend or Resume command and signals interrupt period start/end

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
CMDSUSPEND = 1 and CMDTRANS = 0 Suspend command, start interrupt period when response bit BS = 0.
CMDSUSPEND = 1 and CMDTRANS = 1 Resume command with data, end interrupt period when response bit DF = 1.

Bit 15 **BOOTEN**: Enable boot mode procedure

0: Boot mode procedure disabled
1: Boot mode procedure enabled

Bit 14 **BOOTMODE**: Select the boot mode procedure to be used

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0)
0: Normal boot mode procedure selected
1: Alternative boot mode procedure selected

Bit 13 **DTHOLD**: Hold new data block transmission and reception in the DPSM

If this bit is set, the DPSM does not move from the Wait_S state to the Send state or from the Wait_R state to the Receive state.

Bit 12 **CPSMEN**: Command path state machine (CPSM) enable bit

This bit is written 1 by firmware, and cleared by hardware when the CPSM enters the Idle state.
If this bit is set, the CPSM is enabled.
When DTEN = 1, no command is transfered nor boot procedure is started. CPSMEN is cleared to 0.
During Read Wait with SDMMC_CK stopped no command is sent and CPSMEN is kept 0.

Bit 11 **WAITPEND**: CPSM waits for end of data transfer (CmdPend internal signal) from DPSM

This bit when set, the CPSM waits for the end of data transfer trigger before it starts sending a command.
WAITPEND is only taken into account when DTMODE = *e*•MMC stream data transfer, WIDBUS = 1-bit wide bus mode, DPSMACT = 1 and DTDIR = from host to card.

Bit 10 **WAITINT**: CPSM waits for interrupt request

If this bit is set, the CPSM disables command timeout and waits for an card interrupt request (Response).
If this bit is cleared in the CPSM Wait state, it causes the abort of the interrupt mode.

Bits 9:8 **WAITRESP[1:0]**: Wait for response bits

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
They are used to configure whether the CPSM is to wait for a response, and if yes, which kind of response.
00: No response, expect CMDSENT flag
01: Short response, expect CMDREND or CCRCFAIL flag
10: Short response, expect CMDREND flag (No CRC)
11: Long response, expect CMDREND or CCRCFAIL flag

Bit 7 **CMDSTOP**: The CPSM treats the command as a Stop Transmission command and signals abort to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues the abort signal to the DPSM when the command is sent.

Bit 6 **CMDTRANS**: The CPSM treats the command as a data transfer command, stops the interrupt period, and signals DataEnable to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues an end of interrupt period and issues DataEnable signal to the DPSM when the command is sent.

Bits 5:0 **CMDINDEX[5:0]**: Command index

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

The command index is sent to the card as part of a command message.

Note: 1 At least seven sdmmc_hclk clock periods are needed between two write accesses to this register.

2 MultiMediaCard can send two kinds of response: short responses, 48 bits, or long responses,136 bits. SD card and SD I/O card can send only short responses, the argument can vary according to the type of response: the software distinguishes the type of response according to the send command.

23.10.5 SDMMC command response register (SDMMC_RESPCMDR)

Address offset: 0x010

Reset value: 0x0000 0000

This register contains the command index field of the last command response received. If the command response transmission does not contain the command index field (long or OCR response), the RESPCMD field is unknown, although it must contain 111111b (the value of the reserved field from the response).

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

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

Bits 5:0 **RESPCMD[5:0]**: Response command index

Read-only bitfield. Contains the command index of the last command response received.

23.10.6 SDMMC response x register (SDMMC_RESPxR)

Address offset: 0x010 + 0x004 * x, (x = 1 to 4)

Reset value: 0x0000 0000

These registers contain the status of a card, which is part of the received response.

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

Bits 31:0 **CARDSTATUS[31:0]**: Card status according table below
See *Table 231*.

The card status size is 32 or 128 bits, depending on the response type.

Table 231. Response type and SDMMC_RESPxR registers

Register(1)	Short response	Long response
SDMMC_RESP1R	Card status[31:0]	Card status [127:96]
SDMMC_RESP2R	all 0	Card status [95:64]
SDMMC_RESP3R	all 0	Card status [63:32]
SDMMC_RESP4R	all 0	Card status [31:0](2)

1. The most significant bit of the card status is received first.
2. The SDMMC_RESP4R register LSB is always 0.

23.10.7 SDMMC data timer register (SDMMC_DTIMER)

Address offset: 0x024

Reset value: 0x0000 0000

This register contains the data timeout period, in card bus clock periods.

A counter loads the value from this register, and starts decrementing when the data path state machine (DPSM) enters the Wait_R or Busy state. If the timer reaches 0 while the DPSM is in either of these states, the timeout status flag is set.

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

Bits 31:0 **DATATIME[31:0]**: Data and R1b busy timeout period

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

Data and R1b busy timeout period expressed in card bus clock periods.

Note: A data transfer must be written to the data timer register and the data length register before being written to the data control register.

23.10.8 SDMMC data length register (SDMMC_DLENR)

Address offset: 0x028

Reset value: 0x0000 0000

This register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts.

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

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATALENGTH[24:0]**: Data length value

This register can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Number of data bytes to be transferred.

When DDR = 1 DATALENGTH is truncated to a multiple of 2. (The last odd byte is not transfered)

When DATALENGTH = 0 no data are transfered, when requested by a CPSMEN and CMDTRANS = 1 also no command is transfered. DTEN and CPSMEN are cleared to 0.

Note: For a block data transfer, the value in the data length register must be a multiple of the block size (see SDMMC_DCTRL). A data transfer must be written to the data timer register and the data length register before being written to the data control register.

For an SDMMC multibyte transfer the value in the data length register must be between 1 and 512.

23.10.9 SDMMC data control register (SDMMC_DCTRL)

Address offset: 0x02C

Reset value: 0x0000 0000

This register controls the data path state machine (DPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FIFO RST	BOOT ACK EN	SDIO EN	RW MOD	RW STOP	RW START	DBLOCKSIZE[3:0]				DTMODE[1:0]		DTDIR	DTEN
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **FIFORST**: FIFO reset, flushes any remaining data

This bit can only be written by firmware when IDMAEN= 0 and DPSM is active (DPSMACT = 1). This bit only takes effect when a transfer error or transfer hold occurs.

0: FIFO not affected.

1: Flush any remaining data and reset the FIFO pointers. This bit is automatically cleared to 0 by hardware when DPSM gets inactive (DPSMACT = 0).

Bit 12 **BOOTACKEN**: Enable the reception of the boot acknowledgment

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Boot acknowledgment disabled, not expected to be received

1: Boot acknowledgment enabled, expected to be received

Bit 11 **SDIOEN**: SD I/O interrupt enable functions

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

If this bit is set, the DPSM enables the SD I/O card specific interrupt operation.

Bit 10 **RWMOD**: Read Wait mode

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Read Wait control using SDMMC_D2

1: Read Wait control stopping SDMMC_CK

Bit 9 **RWSTOP**: Read Wait stop

This bit is written by firmware and auto cleared by hardware when the DPSM moves from the R_W state to the Wait_R or Idle state.

0: No Read Wait stop

1: Enable for Read Wait stop when DPSM is in the R_W state.

Bit 8 **RWSTART**: Read Wait start

If this bit is set, Read Wait operation starts.

Bits 7:4 **DBLOCKSIZE[3:0]**: Data block size

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
Define the data block length when the block data transfer mode is selected:
0000: Block length = 2^0 = 1 byte
0001: Block length = 2^1 = 2 bytes
0010: Block length = 2^2 = 4 bytes
0011: Block length = 2^3 = 8 bytes
0100: Block length = 2^4 = 16 bytes
0101: Block length = 2^5 = 32 bytes
0110: Block length = 2^6 = 64 bytes
0111: Block length = 2^7 = 128 bytes
1000: Block length = 2^8 = 256 bytes
1001: Block length = 2^9 = 512 bytes
1010: Block length = 2^{10} = 1024 bytes
1011: Block length = 2^{11} = 2048 bytes
1100: Block length = 2^{12} = 4096 bytes
1101: Block length = 2^{13} = 8192 bytes
1110: Block length = 2^{14} = 16384 bytes
1111: Reserved
When DATALENGTH is not a multiple of DBLOCKSIZE, the transfered data is truncated at a multiple of DBLOCKSIZE. (None of the remaining data are transfered.)
When DDR = 1, DBLOCKSIZE = 0000 must not be used. (No data are transfered)

Bits 3:2 **DTMODE[1:0]**: Data transfer mode selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
00: Block data transfer ending on block count.
01: SDIO multibyte data transfer.
10: *e*•MMC Stream data transfer. (WIDBUS must select 1-bit wide bus mode)
11: Block data transfer ending with STOP_TRANSMISSION command (not to be used with DTEN initiated data transfers).

Bit 1 **DTDIR**: Data transfer direction selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: From host to card.
1: From card to host.

Bit 0 **DTEN**: Data transfer enable bit

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0). This bit is cleared by hardware when data transfer completes.
This bit must only be used to transfer data when no associated data transfer command is used (must not be used with SD or *e*•MMC cards).
0: Do not start data transfer without CPSM data transfer command.
1: Start data transfer without CPSM data transfer command.

23.10.10 SDMMC data counter register (SDMMC_DCNTR)

Address offset: 0x030

Reset value: 0x0000 0000

This register loads the value from the data length register (see SDMMC_DLENR) when the DPSM moves from the Idle state to the Wait_R or Wait_S state. As data is transferred, the counter decrements the value until it reaches 0. The DPSM then moves to the Idle state and

when there has been no error, and no transmit data transfer hold, the data status end flag (DATAEND) is set.

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

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATACOUNT[24:0]**: Data count value

When read, the number of remaining data bytes to be transferred is returned. Write has no effect.

Note: This register must be read only after the data transfer is complete, or hold. When reading after an error event the read data count value may be different from the real number of data bytes transfered.

23.10.11 SDMMC status register (SDMMC_STAR)

Address offset: 0x034

Reset value: 0x0000 0000

This register is a read-only register. It contains two types of flag:

- Static flags (bits [28, 21, 11:0]): these bits remain asserted until they are cleared by writing to the SDMMC interrupt Clear register (see SDMMC_ICR)
- Dynamic flags (bits [20:12]): these bits change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and deasserted as data while written to the FIFO)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTC	IDMA TE	CK STOP	VSW END	ACK TIME OUT	ACK FAIL	SDIOIT	BUSY D0END	BUSY D0	RX FIFOE	TX FIFOE	RX FIFOF	TX FIFOF
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX FIFO HF	TX FIFO HE	CPSM ACT	DPSM ACT	DA BORT	DBCK END	DHOLD	DATA END	CMD SENT	CMDR END	RX OVERR	TX UNDER R	D TIME OUT	C TIME OUT	DCRC FAIL	CCRC FAIL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTC**: IDMA buffer transfer complete

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 27 **IDMATE**: IDMA transfer error

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 26 **CKSTOP**: SDMMC_CK stopped in Voltage switch procedure

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 25 **VSWEND**: Voltage switch critical timing section completion
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 24 **ACKTIMEOUT**: Boot acknowledgment timeout
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 23 **ACKFAIL**: Boot acknowledgment received (boot acknowledgment check fail)
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 22 **SDIOIT**: SDIO interrupt received
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 21 **BUSYD0END**: end of SDMMC_D0 Busy following a CMD response detected
This indicates only end of busy following a CMD response. This bit does not signal busy due to data transfer. Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
0: card SDMMC_D0 signal does NOT signal change from busy to not busy.
1: card SDMMC_D0 signal changed from busy to NOT busy.

Bit 20 **BUSYD0**: Inverted value of SDMMC_D0 line (Busy), sampled at the end of a CMD response and a second time 2 SDMMC_CK cycles after the CMD response
This bit is reset to not busy when the SDMMCD0 line changes from busy to not busy. This bit does not signal busy due to data transfer. This is a hardware status flag only, it does not generate an interrupt.
0: card signals not busy on SDMMC_D0.
1: card signals busy on SDMMC_D0.

Bit 19 **RXFIFOE**: Receive FIFO empty
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes full.

Bit 18 **TXFIFOE**: Transmit FIFO empty
This bit is cleared when one FIFO location becomes full.

Bit 17 **RXFIFOF**: Receive FIFO full
This bit is cleared when one FIFO location becomes empty.

Bit 16 **TXFIFOF**: Transmit FIFO full
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes empty.

Bit 15 **RXFIFOHF**: Receive FIFO half full
There are at least half the number of words in the FIFO. This bit is cleared when the FIFO becomes half+1 empty.

Bit 14 **TXFIFOHE**: Transmit FIFO half empty
At least half the number of words can be written into the FIFO. This bit is cleared when the FIFO becomes half+1 full.

Bit 13 **CPSMACT**: Command path state machine active (not in Idle state)
This is a hardware status flag only, does not generate an interrupt.

Bit 12 **DPSMACT**: Data path state machine active (not in Idle state)
This is a hardware status flag only, does not generate an interrupt.

Bit 11 **DABORT**: Data transfer aborted by CMD12
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 10 **DBCKEND**: Data block sent/received
DBCKEND is set when:
- CRC check passed and DPSM moves to the R_W state
or
- IDMAEN = 0 and transmit data transfer hold and DATACOUNT >0 and DPSM moves to Wait_S.
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 9 **DHOLD**: Data transfer Hold
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 8 **DATAEND**: Data transfer ended correctly
DATAEND is set if data counter DATACOUNT is zero and no errors occur, and no transmit data transfer hold.
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 7 **CMDSENT**: Command sent (no response required)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 6 **CMDREND**: Command response received (CRC check passed, or no CRC)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 5 **RXOVERR**: Received FIFO overrun error (masked by hardware when IDMA is enabled)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 4 **TXUNDERR**: Transmit FIFO underrun error (masked by hardware when IDMA is enabled)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 3 **DTIMEOUT**: Data timeout
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 2 **CTIMEOUT**: Command response timeout
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
The Command Timeout period has a fixed value of 64 SDMMC_CK clock periods.

Bit 1 **DCRCFAIL**: Data block sent/received (CRC check failed)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 0 **CCRCFAIL**: Command response received (CRC check failed)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Note: FIFO interrupt flags must be masked in SDMMC_MASKR when using IDMA mode.

23.10.12 SDMMC interrupt clear register (SDMMC_ICR)

Address offset: 0x038

Reset value: 0x0000 0000

This register is a write-only register. Writing a bit with 1 clears the corresponding bit in the SDMMC_STAR status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCC	IDMA TEC	CK STOPC	VSW ENDC	ACK TIME OUTC	ACK FAILC	SDIO ITC	BUSY D0 ENDC	Res.	Res.	Res.	Res.	Res.
			rw	rw	rw	rw	rw	rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	D ABORT C	DBCK ENDC	DHOLD C	DATA ENDC	CMD SENTC	CMDR ENDC	RX OVERR C	TX UNDER RC	D TIME OUTC	C TIME OUTC	DCRC FAILC	CCRC FAILC
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCC**: IDMA buffer transfer complete clear bit
Set by software to clear the IDMABTC flag.
0: IDMABTC not cleared
1: IDMABTC cleared

Bit 27 **IDMATEC**: IDMA transfer error clear bit
Set by software to clear the IDMATE flag.
0: IDMATE not cleared
1: IDMATE cleared

Bit 26 **CKSTOPC**: CKSTOP flag clear bit
Set by software to clear the CKSTOP flag.
0: CKSTOP not cleared
1: CKSTOP cleared

Bit 25 **VSWENDC**: VSWEND flag clear bit
Set by software to clear the VSWEND flag.
0: VSWEND not cleared
1: VSWEND cleared

Bit 24 **ACKTIMEOUTC**: ACKTIMEOUT flag clear bit
Set by software to clear the ACKTIMEOUT flag.
0: ACKTIMEOUT not cleared
1: ACKTIMEOUT cleared

Bit 23 **ACKFAILC**: ACKFAIL flag clear bit
Set by software to clear the ACKFAIL flag.
0: ACKFAIL not cleared
1: ACKFAIL cleared

Bit 22 **SDIOITC**: SDIOIT flag clear bit
Set by software to clear the SDIOIT flag.
0: SDIOIT not cleared
1: SDIOIT cleared

Bit 21 **BUSYD0ENDC**: BUSYD0END flag clear bit
Set by software to clear the BUSYD0END flag.
0: BUSYD0END not cleared
1: BUSYD0END cleared

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **DABORTC**: DABORT flag clear bit
Set by software to clear the DABORT flag.
0: DABORT not cleared
1: DABORT cleared

Bit 10 **DBCKENDC**: DBCKEND flag clear bit
Set by software to clear the DBCKEND flag.
0: DBCKEND not cleared
1: DBCKEND cleared

Bit 9 **DHOLDC**: DHOLD flag clear bit
Set by software to clear the DHOLD flag.
0: DHOLD not cleared
1: DHOLD cleared

Bit 8 **DATAENDC**: DATAEND flag clear bit
Set by software to clear the DATAEND flag.
0: DATAEND not cleared
1: DATAEND cleared

Bit 7 **CMDSENTC**: CMDSENT flag clear bit
Set by software to clear the CMDSENT flag.
0: CMDSENT not cleared
1: CMDSENT cleared

Bit 6 **CMDRENDC**: CMDREND flag clear bit
Set by software to clear the CMDREND flag.
0: CMDREND not cleared
1: CMDREND cleared

Bit 5 **RXOVERRC**: RXOVERR flag clear bit
Set by software to clear the RXOVERR flag.
0: RXOVERR not cleared
1: RXOVERR cleared

Bit 4 **TXUNDERRC**: TXUNDERR flag clear bit
Set by software to clear TXUNDERR flag.
0: TXUNDERR not cleared
1: TXUNDERR cleared

Bit 3 **DTIMEOUTC**: DTIMEOUT flag clear bit
Set by software to clear the DTIMEOUT flag.
0: DTIMEOUT not cleared
1: DTIMEOUT cleared

Bit 2 **CTIMEOUTC**: CTIMEOUT flag clear bit
Set by software to clear the CTIMEOUT flag.
0: CTIMEOUT not cleared
1: CTIMEOUT cleared

Bit 1 **DCRCFAILC**: DCRCFAIL flag clear bit
Set by software to clear the DCRCFAIL flag.
0: DCRCFAIL not cleared
1: DCRCFAIL cleared

Bit 0 **CCRCFAILC**: CCRCFAIL flag clear bit
Set by software to clear the CCRCFAIL flag.
0: CCRCFAIL not cleared
1: CCRCFAIL cleared

23.10.13 SDMMC mask register (SDMMC_MASKR)

Address offset: 0x03C

Reset value: 0x0000 0000

This register determines which status flags generate an interrupt request by setting the corresponding bit to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCIE	Res.	CK STOP IE	VSW ENDIE	ACK TIME OUTIE	ACK FAILIE	SDIO ITIE	BUSY D0 ENDIE	Res.	Res.	TX FIFO EIE	RX FIFO FIE	Res.
			rw		rw	rw	rw	rw	rw	rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX FIFO HFIE	TX FIFO HEIE	Res.	Res.	DA BORT IE	DBCK ENDIE	DHOLD IE	DATA ENDIE	CMD SENT IE	CMDR ENDIE	RX OVER RIE	TX UNDER RIE	D TIME OUTIE	C TIME OUTIE	DCRC FAILIE	CCRC FAILIE
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCIE**: IDMA buffer transfer complete interrupt enable
Set and cleared by software to enable/disable the interrupt generated when the IDMA has transferred all data belonging to a memory buffer.
0: IDMA buffer transfer complete interrupt disabled
1: IDMA buffer transfer complete interrupt enabled

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CKSTOPIE**: Voltage switch clock stopped interrupt enable
Set and cleared by software to enable/disable interrupt caused by voltage switch clock stopped.
0: Voltage switch clock stopped interrupt disabled
1: Voltage switch clock stopped interrupt enabled

Bit 25 **VSWENDIE**: Voltage switch critical timing section completion interrupt enable
Set and cleared by software to enable/disable the interrupt generated when voltage switch critical timing section completion.
0: Voltage switch critical timing section completion interrupt disabled
1: Voltage switch critical timing section completion interrupt enabled

Bit 24 **ACKTIMEOUTIE**: Acknowledgment timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment timeout.
0: Acknowledgment timeout interrupt disabled
1: Acknowledgment timeout interrupt enabled

Bit 23 **ACKFAILIE**: Acknowledgment fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment fail.
0: Acknowledgment fail interrupt disabled
1: Acknowledgment fail interrupt enabled

Bit 22 **SDIOITIE**: SDIO mode interrupt received interrupt enable
Set and cleared by software to enable/disable the interrupt generated when receiving the SDIO mode interrupt.
0: SDIO mode interrupt received interrupt disabled
1: SDIO mode interrupt received interrupt enabled

Bit 21 **BUSYD0ENDIE**: BUSYD0END interrupt enable
Set and cleared by software to enable/disable the interrupt generated when SDMMC_D0 signal changes from busy to NOT busy following a CMD response.
0: BUSYD0END interrupt disabled
1: BUSYD0END interrupt enabled

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TXFIFOEIE**: Tx FIFO empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO empty.
0: Tx FIFO empty interrupt disabled
1: Tx FIFO empty interrupt enabled

Bit 17 **RXFIFOFIE**: Rx FIFO full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO full.
0: Rx FIFO full interrupt disabled
1: Rx FIFO full interrupt enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **RXFIFOHFIE**: Rx FIFO half full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO half full.
0: Rx FIFO half full interrupt disabled
1: Rx FIFO half full interrupt enabled

Bit 14 **TXFIFOHEIE**: Tx FIFO half empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO half empty.
0: Tx FIFO half empty interrupt disabled
1: Tx FIFO half empty interrupt enabled

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **DABORTIE**: Data transfer aborted interrupt enable
Set and cleared by software to enable/disable interrupt caused by a data transfer being aborted.
0: Data transfer abort interrupt disabled
1: Data transfer abort interrupt enabled

Bit 10 **DBCKENDIE**: Data block end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data block end.
0: Data block end interrupt disabled
1: Data block end interrupt enabled

Bit 9 **DHOLDIE**: Data hold interrupt enable
Set and cleared by software to enable/disable the interrupt generated when sending new data is hold in the DPSM Wait_S state.
0: Data hold interrupt disabled
1: Data hold interrupt enabled

Bit 8 **DATAENDIE**: Data end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data end.
0: Data end interrupt disabled
1: Data end interrupt enabled

Bit 7 **CMDSENTIE**: Command sent interrupt enable
Set and cleared by software to enable/disable interrupt caused by sending command.
0: Command sent interrupt disabled
1: Command sent interrupt enabled

Bit 6 **CMDRENDIE**: Command response received interrupt enable
Set and cleared by software to enable/disable interrupt caused by receiving command response.
0: Command response received interrupt disabled
1: command Response received interrupt enabled

Bit 5 **RXOVERRIE**: Rx FIFO overrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO overrun error.
0: Rx FIFO overrun error interrupt disabled
1: Rx FIFO overrun error interrupt enabled

Bit 4 **TXUNDERRIE**: Tx FIFO underrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO underrun error.
0: Tx FIFO underrun error interrupt disabled
1: Tx FIFO underrun error interrupt enabled

Bit 3 **DTIMEOUTIE**: Data timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by data timeout.
0: Data timeout interrupt disabled
1: Data timeout interrupt enabled

Bit 2 **CTIMEOUTIE**: Command timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by command timeout.
0: Command timeout interrupt disabled
1: Command timeout interrupt enabled

Bit 1 **DCRCFAILIE**: Data CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by data CRC failure.
0: Data CRC fail interrupt disabled
1: Data CRC fail interrupt enabled

Bit 0 **CCRCFAILIE**: Command CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by command CRC failure.
0: Command CRC fail interrupt disabled
1: Command CRC fail interrupt enabled

23.10.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER)

Address offset: 0x040

Reset value: 0x0000 0000

This register contains the acknowledgment timeout period, in SDMMC_CK bus clock periods.

A counter loads the value from this register, and starts decrementing when the data path state machine (DPSM) enters the Wait_Ack state. If the timer reaches 0 while the DPSM is in this states, the acknowledgment timeout status flag is set.

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

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **ACKTIME[24:0]**: Boot acknowledgment timeout period
This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
Boot acknowledgment timeout period expressed in card bus clock periods.

Note: The data transfer must be written to the acknowledgment timer register before being written to the data control register.

23.10.15 SDMMC DMA control register (SDMMC_IDMACTRLR)

Address offset: 0x050

Reset value: 0x0000 0000

The receive and transmit FIFOs can be read or written as 32-bit wide registers. The FIFOs contain 32 entries on 32 sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO.

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

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

Bit 1 **IDMABMODE**: Buffer mode selection
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: Single buffer mode.
1: Linked list mode.

Bit 0 **IDMAEN**: IDMA enable
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: IDMA disabled
1: IDMA enabled

23.10.16 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER)

Address offset: 0x054

Reset value: 0x0000 0000

This register contains the buffer size when in linked list configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABNDT[11]
															rw

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

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:5 **IDMABNDT[11:0]**: Number of bytes per buffer
This 12-bit value must be multiplied by 8 to get the size of the buffer in 32-bit words and by 32 to get the size of the buffer in bytes.
Example: IDMABNDT = 0x001: buffer size = 8 words = 32 bytes.
Example: IDMABNDT = 0x800: buffer size = 16384 words = 64 Kbyte.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 4:0 Reserved, must be kept at reset value.

23.10.17 SDMMC IDMA buffer base address register (SDMMC_IDMABASER)

Address offset: 0x058

Reset value: 0x0000 0000

This register contains the memory buffer base address in single buffer configuration and linked list configuration.

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

Bits 31:0 **IDMABASE[31:0]**: Buffer memory base address bits [31:2], must be word aligned (bit [1:0] are always 0 and read only)

This register can be written by firmware when DPSM is inactive (DPSMACT = 0), and can dynamically be written by firmware when DPSM active (DPSMACT = 1).

23.10.18 SDMMC IDMA linked list address register (SDMMC_IDMALAR)

Address offset: 0x064

Reset value: 0x0000 0000

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

Bit 31 **ULA**: Update SDMMC_IDMALAR from linked list when in linked list mode (SDMMC_IDMACTRLR.IDMABMODE select linked list mode)
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: SDMMC_IDMALAR is not to be updated, last linked list item.
1: SDMMC_IDMALAR is to be updated from linked list table.

Bit 30 **ULS**: Update SDMMC_IDMABSIZE from the next linked list when in linked list mode (SDMMC_IDMACTRLR.IDMABMODE select linked list mode and ULA = 1)
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: SDMMC_IDMABSIZER is not to be updated from next linked list table.
1: SDMMC_IDMABSIZER is to be updated from next linked list table.

Bit 29 **ABR**: Acknowledge linked list buffer ready
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
This bit is not taken into account when starting the first linked list buffer from the software programmed register information. ABR is only taken into account on subsequent loaded linked list items.
0: Loaded linked list buffer is not ready (this causes a linked list IDMA transfer error to be generated).
1: Loaded linked list buffer ready acknowledge. Linked list buffer data are transfered by IDMA.

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:2 **IDMALA[13:0]**: Word aligned linked list item address offset
Linked list item offset pointer to the base of the next linked list item structure.
Linked list item base address is IDMABA + IDMALA.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

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

23.10.19 SDMMC IDMA linked list memory base register (SDMMC_IDMABAR)

Address offset: 0x068

Reset value: 0x0000 0000

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

Bits 31:2 **IDMABA[29:0]**: Word aligned Linked list memory base address
Linked list memory base pointer.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

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

23.10.20 SDMMC data FIFO registers x (SDMMC_FIFORx)

Address offset: 0x080 + 0x004 * x, (x =0 to 15)

Reset value: 0x0000 0000

The receive and transmit FIFOs can be only read or written as word (32-bit) wide registers. The FIFOs contain 16 entries on sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO. The FIFO register interface takes care of correct data alignment inside the FIFO, the FIFO register address used by the CPU does matter.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.

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

Bits 31:0 **FIFODATA[31:0]**: Receive and transmit FIFO data

This register can only be read or written by firmware when the DPSM is active (DPSMACT = 1).

The FIFO data occupies 16 entries of 32-bit words.

23.10.21 SDMMC register map

Table 232. SDMMC register map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SDMMC_ POWER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRPOL	VSWITCHEN	VSWITCH	PWRCTRL[1:0]	
	Reset value																												0	0	0	0	0
0x004	SDMMC_ CLKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]		BUSSPEED	DDR	HWFC_EN	NEGEDGE	WIDBUS[1:0]		Res.	PWRSAV	Res.	Res.	CLKDIV[9:0]									
	Reset value											0	0	0	0	0	0	0	0		0			0	0	0	0	0	0	0	0	0	0
0x008	SDMMC_ ARGR	CMDARG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	SDMMC_ CMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDSUSPEND	BOOTEN	BOOTMODE	DTHOLD	CPSMEN	WAITPEND	WAITINT	WAITRESP[1:0]		CMDSTOP	CMDTRANS	CMDINDEX[5:0]					
	Reset value																																

Table 232. SDMMC register map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	SDMMC_ RESPCMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESPCMD[5:0]					
	Reset value																											0	0	0	0	0	0
0x014	SDMMC_ RESP1R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	SDMMC_ RESP2R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	SDMMC_ RESP3R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	SDMMC_ RESP4R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	SDMMC_ DTIMER	DATATIME[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	SDMMC_ DLENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATALENGTH[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	SDMMC_ DCTRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFORST	BOOTACKEN	SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCK SIZE[3:0]				DTMODE[1:0]		DTDIR	DTEN
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	SDMMC_ DCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATACOUNT[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	SDMMC_ STAR	Res.	Res.	Res.	IDMABTC	IDMATE	CKSTOP	VSWEND	ACKTIMEOUT	ACKFAIL	SDIOIT	BUSYD0END	BUSYD0	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	CPSMACT	DPSMACT	DABORT	DBCKEND	DHOLD	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	SDMMC_ ICR	Res.	Res.	Res.	IDMABTCC	IDMATEC	CKSTOPC	VSWENDC	ACKTIMEOUTC	ACKFAILC	SDIOITC	BUSYD0ENDC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DABORTC	DBCKENDC	DHOLDC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
	Reset value				0	0	0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0
0x03C	SDMMC_ MASKR	Res.	Res.	Res.	IDMABTCIE	Res.	CKSTOPIE	VSWENDIE	ACKTIMEOUTIE	ACKFAILIE	SDIOITIE	BUSYD0ENDIE	Res.	Res.	TXFIFOEIE	RXFIFOFIE	Res.	RXFIFOHFIE	TXFIFOHEIE	Res.	Res.	DABORTIE	DBCKENDIE	DHOLDIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
	Reset value				0		0	0	0	0	0	0			0	0		0	0			0	0	0	0	0	0	0	0	0	0	0	0

Table 232. SDMMC register map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	SDMMC_ ACKTIMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACKTIME[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044 - 0x04C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x050	SDMMC_ IDMACTRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABMODE	IDMAEN
	Reset value																															0	0
0x054	SDMMC_ IDMABSIZER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABNDT[11:0]												Res.	Res.	Res.	Res.	Res.
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0					
0x058	SDMMC_ IDMABASER	IDMABASE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C - 0x060	Reserved	Res.																															
0x064	SDMMC_ IDMALAR	ULA	ULS	ABR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMALA[13:0]														Res.	Res.
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0			
0x068	SDMMC_ IDMABAR	IDMABA[29:0]																														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x06C - 0x07C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x080 + 0x04 * x, (x=0..15)	SDMMC_ FIFORx	FIFODATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

24 Delay block (DLYB)

24.1 DLYB introduction

The delay block (DLYB) is used to generate an output clock that is dephased from the input clock. The phase of the output clock must be programmed by the user application. The output clock is then used to clock the data received by another peripheral such as an SDMMC or XSPI interface.

The delay is voltage- and temperature-dependent, that may require the application to re-configure and recenter the output clock phase with the receive data.

24.2 DLYB main features

The delay block has the following features:

- Input clock frequency ranging from 25 MHz to the maximum frequency supported by the communication interface (see datasheet)
- Up to 12 oversampling phases.

24.3 DLYB implementation

Table 233. STM32H7Pxxx features

DLYB associated peripheral	DLYBXS1	DLYBSD1
XSPI1	X	-
SDMMC1	-	X

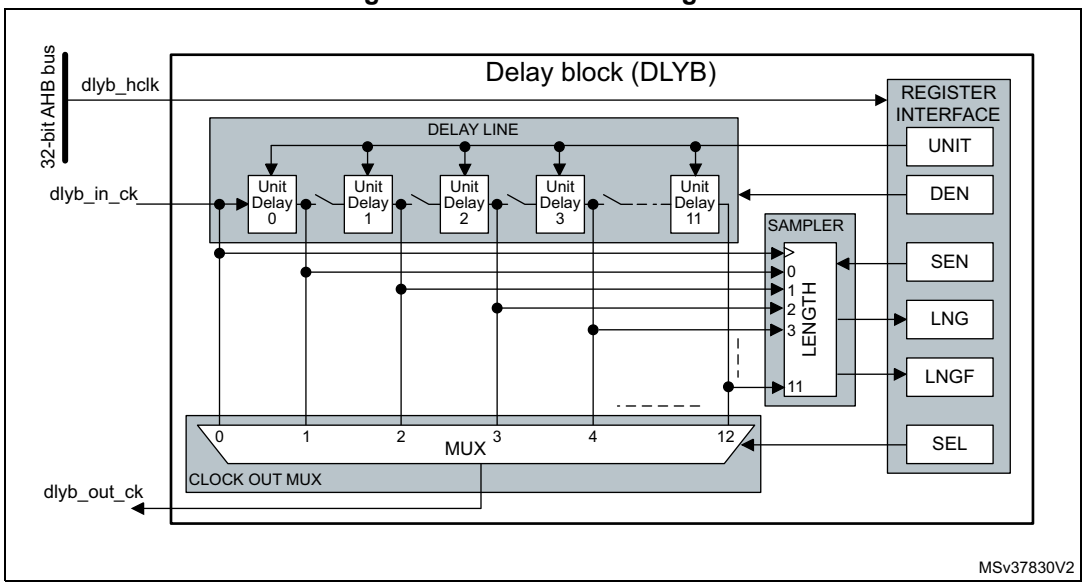
24.4 DLYB functional description

24.4.1 DLYB diagram

The delay block includes the following sub-blocks (shown in the figure below):

- register interface block providing AHB access to the DLYB registers
- delay line supporting the unit delays
- delay line length sampling
- output clock selection multiplexer

Figure 282. DLYB block diagram



24.4.2 DLYB pins and internal signals

Table 234 lists the DLYB internal signals.

Table 234. DLYB internal input/output signals

Signal name	Signal type	Description
dlyb_hclk	Digital input	Delay block register interface clock
dlyb_in_ck	Digital input	Delay block input clock
dlyb_out_ck	Digital output	Delay block output clock

Table 235. DLYB interconnection

Signal name	Source/destination	
	SDMMC1	XSPI1
dlyb_in_ck	sdmmc_io_in_ck	DQS
dlyb_out_ck	sdmmc_fb_ck	DQS delayed for data sampling

24.4.3 General description

The delay block is enabled by setting the DEN bit in the DLYB control register (DLYB_CR). The length sampler is enabled through the SEN bit in DLYB_CR register.

When the delay block is enabled, the delay added by a unit delay is defined by the UNIT[6:0] field in the DLYB configuration register (DLYB_CFGR).

Note: UNIT[6:0] can be programmed only when the output clock is disabled (SEN = 1).

When the delay block is enabled, the output clock phase is selected through the SEL[3:0] field in DLYB_CFGR register.

Note: SEL can be programmed only when the output clock is disabled (SEN = 1).

The output clock can be de-phased over one input clock period by configuring the delay line length to span one period. The delay line length can be configured by enabling the length sampler through the SEN bit, that gives access to the delay line length (LNG[11:0]) and length valid flag (LNGF) in DLYB_CFGR.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced. This allows a smaller unit delay providing higher resolution.

Once the delay line length is configured, a dephased output clock can be selected by the output clock multiplexer. This is done through SEL[3:0]. The output clock is only available on the selected phase when SEN is set to 0.

The table below gives a summary of the delay block control.

Table 236. Delay block control

DEN	SEN	UNIT	SEL	LNG	LNGF	Output clock
0	0	Don’t care	Don’t care	Don’t care	Don’t care	Enabled (= Input clock)
x	1	Unit delay	Output clock phase	Length	Length flag	Disabled
1	0	Unit delay(1)	Output clock phase(2)	Don’t care	Don’t care	Enabled (= selected phase)

1. The unit delay can only be changed when SEN = 1.
2. The output clock phase can only be changed when SEN = 1.

24.4.4 Delay line length configuration procedure

LNG[11:0] is used to determine the delay line length with respect to the input clock period. The length must be configured so that one full input clock period is covered by the delay line length.

Note that despite the delay line has 12 unit delay elements, the following procedure description returns a length between 0 and 10, as the upper delay output value is used to ensure that the delay is calibrated over one full input clock cycle. Depending on the clock frequency and UNIT value, unit delay element 10 may also be truncated from the clock cycle length.

A clock input (free running clock) must be present during the whole tuning procedure.

To configure the delay line length to one period of the Input clock, follow the sequence below:

1. Enable the delay block by setting DEN bit to 1.
2. Enable the length sampling by setting SEN bit to 1.
3. Enable all delay cells by setting SEL[3:0] to 12.
4. For UNIT[6:0] = 0 to 127 (this step must be repeated until the delay line length is configured):
   a) Update the UNIT[6:0] value and wait till the length flag LNGF is set to 1.
   b) Read LNG[11:0].

   If (LNG[10:0] > 0) and (LNG[11] or LNG[10] = 0), the delay line length is configured to one input clock period.
5. Determine how many unit delays (N) span one input clock period: for N = 0 to 10, if LNG[N] = 1, the number of unit delays spanning the input clock period = N.
6. Disable the length sampling by clearing SEN to 0.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced smaller than one input clock period. This allows a smaller unit delay, providing a higher resolution spanning a shorter time interval.

24.4.5 Output clock phase configuration procedure

When the delay line length is configured to one input clock period, the output clock phase can be selected between the unit delays spanning one Input clock period.

Follow the steps below to select the output clock phase:

1. Disable the output clock and enable the access to the phase selection SEL[3:0] bits by setting SEN bit to 1.
2. Program SEL[3:0] with the desired output clock phase value.
3. Enable the output clock on the selected phase by clearing SEN to 0.

XSPI use case:

The delay block is used in conjunction with XSPI interface to allow shifting the input data sampling signal. This sampling signal can be the feedback clock or the data strobe (DQS) signal, which is delivered by certain type of devices. Note that in case DQS is used, the calibration procedure must be performed beforehand with a free running clock, as DQS is a discontinuous signal.

In case of SDR (single data rate) mode the user must typically shift the sampling signal by half period, so that the sampling edges are positioned in the middle of the valid data phase.

In case of DDR (dual data rate) mode, for which data are transitioning at start and middle of period, typical value must be close to N/4, once the calibration is completed.

In case of high frequencies and tight timing constraints, the delay setting granularity (10) might be too coarse. Since in most cases it is not necessary to have a possible delay value covering the whole sampling clock period, the "Unit" value can be overridden by application in order to improve the accuracy of the sampling edge position (example: providing a twice as small “Unit” gives twice better timing accuracy. The counterpart being that the maximum possible delay is divided by 2).

SDMMC use case:

The delay block is used in conjunction with SDMMC interface variable delay. For correct sampling point tuning the delay value must cover a whole SDMMC_CK clock period. After having tuned the delay line length the individual delays are used in the sampling point tuning to find the optimal sampling point.

24.5 DLYB registers

All registers can be accessed in word, half-word and byte access.

24.5.1 DLYB control register (DLYB_CR)

Address offset: 0x000

Reset value: 0x0000 0000

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

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

Bit 1 **SEN:** Sampler length enable bit

0: Sampler length and register access to UNIT[6:0] and SEL[3:0] disabled, output clock enabled.

1: Sampler length and register access to UNIT[6:0] and SEL[3:0] enabled, output clock disabled.

Bit 0 **DEN:** Delay block enable bit

0: DLYB disabled.

1: DLYB enabled.

24.5.2 DLYB configuration register (DLYB_CFGR)

Address offset: 0x004

Reset value: 0x0000 0000

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

Bit 31 **LNGF:** Length valid flag
This flag indicates when the delay line length value contained in LNG[11:0] is valid after UNIT[6:0] bits changed.
0: Length value in LNG is not valid.
1: Length value in LNG is valid.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **LNG[11:0]:** Delay line length value
These bits reflect the 12 unit delay values sampled at the rising edge of the input clock.
The value is only valid when LNGF = 1.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **UNIT[6:0]:** Delay of a unit delay cell.
These bits can only be written when SEN = 1.
Unit delay = initial delay + UNIT[6:0] x delay step

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL[3:0]:** Phase for the output clock.
These bits can only be written when SEN = 1.
Output clock phase = input clock + SEL[3:0] x unit delay

24.5.3 DLYB register map

Table 237. DLYB register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DLYB_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEN	DEN
	Reset value																															0	0
0x004	DLYB_CFGR	LNGF	Res.	Res.	Res.	LNG[11:0]												Res.	UNIT[6:0]							Res.	Res.	Res.	Res.	SEL[3:0]			
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0					0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

25 Analog-to-digital converters (ADC)

25.1 ADC introduction

This section describes the implementation of up to 4 ADCs:

- ADC1 and ADC2 are tightly coupled and can operate in dual mode (ADC1 is master).
- ADC3 and ADC4 are tightly coupled and can operate in dual mode (ADC3 is master).

Each ADC consists of one 12-bit successive approximation analog-to-digital converter.

Each ADC has up to 20 multiplexed channels. A/D conversion of the various channels can be performed in single, continuous, scan, or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned (default configuration) 32-bit data register.

The ADCs are mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows improving analog performances while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

25.2 ADC main features

- High-performance features
  - Up to 4 ADCs: ADC1/2 and ADC3/4 can operate in dual mode
    ADC1 is connected to 16 external channels + 4 internal channels
    ADC2 is connected to 18 external channels + 2 internal channels
    ADC3 is connected to 17 external channels + 3 internal channels
    ADC4 is connected to 18 external channels + 2 internal channels
  - 12, 10, 8 or 6-bit configurable resolution
  - ADC conversion time independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - Management of single-ended, pseudodifferential, and differential inputs (programmable per channels)
  - AHB slave bus interface to allow fast data handling
  - Channel-wise programmable sampling time
  - Flexible sampling time control
  - Fixed latency for a trigger to start of sampling
  - Up to 4 injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Hardware assistant to prepare the context of the injected channels to allow fast context switching
  - Data alignment with in-built data coherency
  - Data can be managed by DMA for regular channel conversions
  - Data can be routed to ADF for post processing
  - Four dedicated data registers for the injected channels
- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
  - Allows slow bus frequency application while keeping optimum ADC performance
  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency application (autodelayed mode)
- Oversampler
  - 32-bit data register
  - Oversampling ratio adjustable from 2 to 1024
  - Programmable data right and left shifts
  - Dedicated accumulator for each regular and injected channel
  - Parallel oversampling mode for injected channels (access to the intermediate oversampling result)
- Data preconditioning
  - Gain compensation
  - Offset compensation
- Analog input channels
  - External analog inputs (per ADC): up to 18 GPIO pads

  - 4 channels for the internal reference voltage (V_{REFINT})
  - 1 channel for the internal temperature sensor (V_{SENSE})
  - 1 channel for monitoring the external VBAT power supply pin
  - 1 channel for monitoring V_{DDCORE} internal voltage
  - Connection to DAC internal channels
- Start-of-conversion can be initiated:
  - By software for both regular and injected conversions
  - By hardware triggers with configurable polarity (internal timer events or GPIO input events) for both regular and injected conversions
- Conversion modes
  - Each ADC can convert a single channel or can scan a sequence of channels
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
- 3 analog watchdogs per ADC, with filters
- ADC input range: $V_{SSA} \leq V_{IN} \leq V_{REF+}$

Figure 283 shows the block diagram of one ADC.

25.3 ADC implementation

Table 238. ADC features

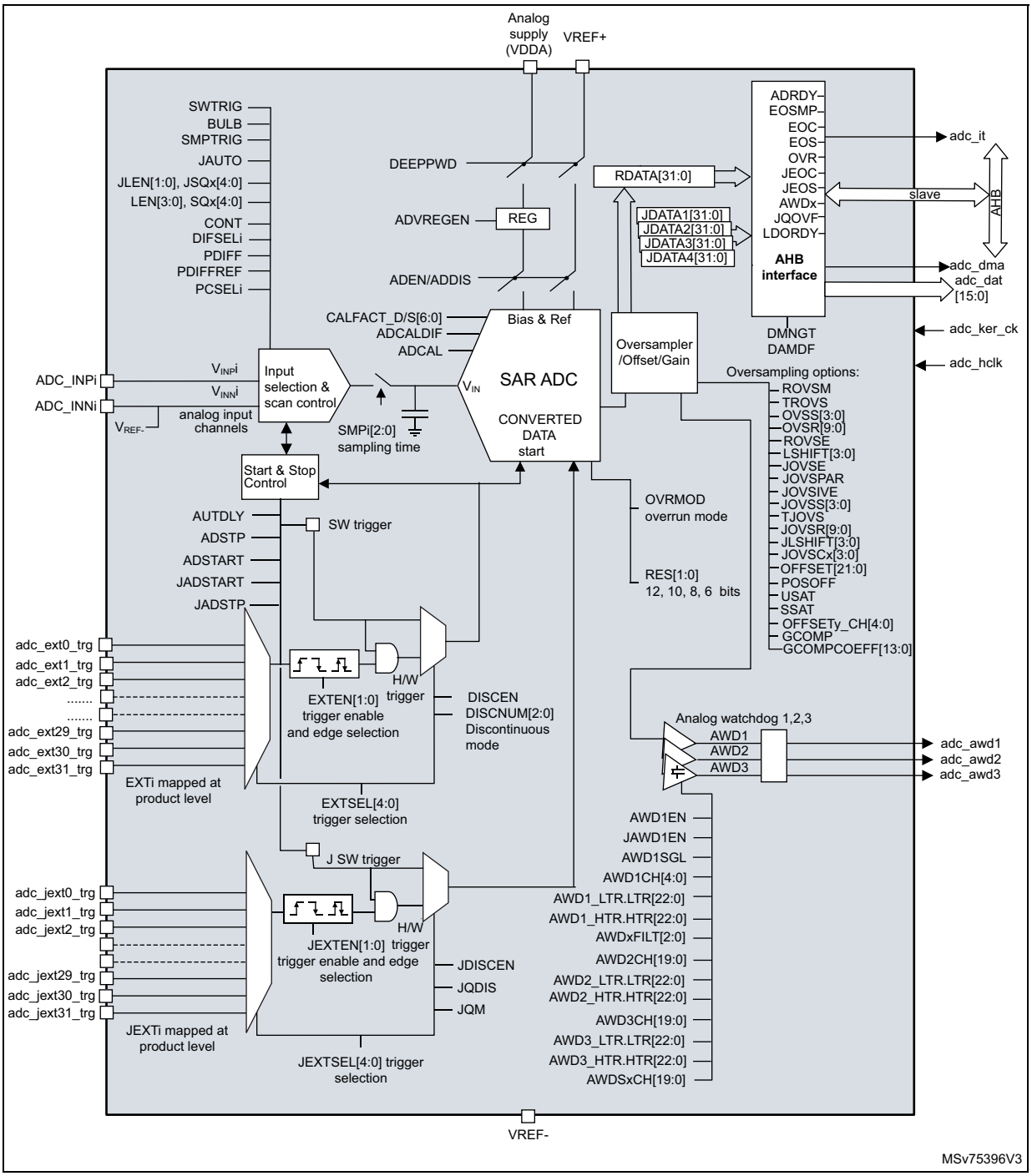
ADC modes/features	ADC1, ADC2	ADC3, ADC4
Resolution	12 bits	
Maximum sampling speed	5 Msps (12-bit resolution)	
Dual mode operation	X	
Offset calibration	X	
Single-end input	X	
Differential input	X	
Injected channel conversion	X	
Oversampling	up to x1024	
Data register	32 bits	
DMA support	X	
Parallel data output to ADF	X	-
Offset compensation	X	
Gain compensation	X	
Number of analog watchdog	3	

25.4 ADC functional description

25.4.1 ADC block diagram

Figure 283 shows the ADC block diagram and *Table 239* gives the ADC pin description.

Figure 283. ADC block diagram



25.4.2 ADC pins and internal signals

Table 239. ADC input/output pins

Pin name	Signal type	Description
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply, equal to V_{SS}.
VREF+	Input, analog reference positive	The higher/positive reference voltage for the ADC.
VREF-	Input, analog reference negative	The lower/negative reference voltage for the ADC.
ADCx_INNi/INPi	Negative/positive external analog input signals	20 negative/positive external analog input channels (refer to *Section 25.4.4: ADC connectivity* for details).

Table 240. ADC internal input/output signals

Internal signal name	Signal type	Description
$V_{INP}i$	Positive analog input channels	Positive internal analog input channels connected either to ADCx_INP*i* external channels or to internal channels.
$V_{INN}i$	Negative analog input channels	Negative internal analog input channels connected either to ADCx_INN*i* external channels or to internal channels.
adc_ext_trgi	Inputs	ADC external trigger inputs for regular conversions.These inputs are shared between the ADC master and the ADC slave.
adc_jext_trgi	Inputs	ADC external trigger inputs for the injected conversions. These inputs are shared between the ADC master and the ADC slave.
adc_awdy	Output	Internal analog watchdog output signal connected to on-chip timers. (y = Analog watchdog number 1,2,3).
adc_ker_ck	Input	ADC kernel clock
adc_hclk	Input	ADC peripheral clock
adc_it	Output	ADC interrupt
adc_dma	Output	ADC DMA request
adc_dat[15:0]	Output	ADC data outputs

Table 241. ADC1/2/3/4 interconnection

Signal name	Source/destination
ADC1 V_{INP}[16]	V_{SENSE}
ADC1 V_{INP}[17]	V_{REFINT}
ADC1 V_{INP}[18]	V_{BAT} / 4
ADC1 V_{INP}[19]	dac1_out1

Table 241. ADC1/2/3/4 interconnection (continued)

Signal name	Source/destination
ADC1 V_{INN}[19]	dac1_out2
ADC2 V_{INP}[18]	V_{REFINT}
ADC2 V_{INP}[19]	dac1_out1
ADC2 V_{INN}[19]	dac1_out2
ADC3 V_{INP}[17]	V_{DDCORE}
ADC3 V_{INP}[18]	V_{REFINT}
ADC3 V_{INP}[19]	dac1_out1
ADC3 V_{INN}[19]	dac2_out1
ADC4 V_{INP}[18]	V_{REFINT}
ADC4 V_{INP}[19]	dac1_out1
ADC4 V_{INN}[19]	dac1_out2
adc1_dat[15:0]	adf_adc1_dat[15:0]
adc2_dat[15:0]	adf_adc2_dat[15:0]
adc1_ext_trg0, adc2_ext_trg0	exti11
adc1_ext_trg1, adc2_ext_trg1	tim1_oc1
adc1_ext_trg2, adc2_ext_trg2	tim1_oc2
adc1_ext_trg3, adc2_ext_trg3	tim1_oc3
adc1_ext_trg4, adc2_ext_trg4	tim1_trgo
adc1_ext_trg5, adc2_ext_trg5	tim1_trgo2
adc1_ext_trg6, adc2_ext_trg6	tim2_oc2
adc1_ext_trg7, adc2_ext_trg7	tim2_trgo
adc1_ext_trg8, adc2_ext_trg8	tim3_oc4 / tm4_oc4(1)
adc1_ext_trg9, adc2_ext_trg9	tim3_trgo / tim4_trgo(1)
adc1_ext_trg10, adc2_ext_trg10	tim6_trgo
adc1_ext_trg11, adc2_ext_trg11	tim8_trgo
adc1_ext_trg12, adc2_ext_trg12	tim8_trgo2
adc1_ext_trg13, adc2_ext_trg13	tim9_trgo
adc1_ext_trg14, adc2_ext_trg14	tim15_trgo
adc1_ext_trg15, adc2_ext_trg15	tim20_oc1
adc1_ext_trg16, adc2_ext_trg16	tim20_oc2
adc1_ext_trg17, adc2_ext_trg17	tim20_trgo
adc1_ext_trg18, adc2_ext_trg18	tim20_trgo2
adc1_ext_trg19, adc2_ext_trg19	lptim1_ch1
adc1_ext_trg20, adc2_ext_trg20	hrtim1_adc_trg1
adc1_ext_trg21, adc2_ext_trg21	hrtim1_adc_trg2

Table 241. ADC1/2/3/4 interconnection (continued)

Signal name	Source/destination
adc1_ext_trg22, adc2_ext_trg22	hrtim1_adc_trg3
adc1_ext_trg23, adc2_ext_trg23	hrtim1_adc_trg4
adc1_ext_trg24, adc2_ext_trg24	hrtim1_adc_trg5
adc1_ext_trg25, adc2_ext_trg25	hrtim1_adc_trg6
adc1_ext_trg26, adc2_ext_trg26	hrtim2_adc_trg1
adc1_ext_trg27, adc2_ext_trg27	hrtim2_adc_trg2
adc1_ext_trg28, adc2_ext_trg28	hrtim2_adc_trg3
adc1_ext_trg29, adc2_ext_trg29	hrtim2_adc_trg4
adc1_ext_trg30, adc2_ext_trg30	hrtim2_adc_trg5
adc1_ext_trg31, adc2_ext_trg31	hrtim2_adc_trg6
adc1_jext_trg0, adc2_jext_trg0	exti15
adc1_jext_trg1, adc2_jext_trg1	tim1_oc4
adc1_jext_trg2, adc2_jext_trg2	tim1_trgo
adc1_jext_trg3, adc2_jext_trg3	tim1_trgo2
adc1_jext_trg4, adc2_jext_trg4	tim2_oc1
adc1_jext_trg5, adc2_jext_trg5	tim2_trgo
adc1_jext_trg6, adc2_jext_trg6	tim3_oc1
adc1_jext_trg7, adc2_jext_trg7	tim3_oc3
adc1_jext_trg8, adc2_jext_trg8	tim3_oc4
adc1_jext_trg9, adc2_jext_trg9	tim3_trgo / tim4_trgo(1)
adc1_jext_trg10, adc2_jext_trg10	tim7_trgo
adc1_jext_trg11, adc2_jext_trg11	tim8_oc4
adc1_jext_trg12, adc2_jext_trg12	tim8_trgo
adc1_jext_trg13, adc2_jext_trg13	tim8_trgo2
adc1_jext_trg14, adc2_jext_trg14	tim12_trgo
adc1_jext_trg15, adc2_jext_trg15	tim15_trgo
adc1_jext_trg16, adc2_jext_trg16	Reserved
adc1_jext_trg17, adc2_jext_trg17	tim20_trgo
adc1_jext_trg18, adc2_jext_trg18	tim20_trgo2
adc1_jext_trg19, adc2_jext_trg19	lptim1_ch2
adc1_jext_trg20, adc2_jext_trg20	hrtim1_adc_trg1
adc1_jext_trg21, adc2_jext_trg21	hrtim1_adc_trg2
adc1_jext_trg22, adc2_jext_trg22	hrtim1_adc_trg3
adc1_jext_trg23, adc2_jext_trg23	hrtim1_adc_trg4
adc1_jext_trg24, adc2_jext_trg24	hrtim1_adc_trg5

Table 241. ADC1/2/3/4 interconnection (continued)

Signal name	Source/destination
adc1_jext_trg25, adc2_jext_trg25	hrtim1_adc_trg6
adc1_jext_trg26, adc2_jext_trg26	hrtim2_adc_trg1
adc1_jext_trg27, adc2_jext_trg27	hrtim2_adc_trg2
adc1_jext_trg28, adc2_jext_trg28	hrtim2_adc_trg3
adc1_jext_trg29, adc2_jext_trg29	hrtim2_adc_trg4
adc1_jext_trg30, adc2_jext_trg30	hrtim2_adc_trg5
adc1_jext_trg31, adc2_jext_trg31	hrtim2_adc_trg6
adc3_ext_trg0, adc4_ext_trg0	exti2
adc3_ext_trg1, adc4_ext_trg1	tim1_oc3
adc3_ext_trg2, adc4_ext_trg2	tim1_trgo
adc3_ext_trg3, adc4_ext_trg3	tim1_trgo2
adc3_ext_trg4, adc4_ext_trg4	tim2_oc1
adc3_ext_trg5, adc4_ext_trg5	tim2_oc3
adc3_ext_trg6, adc4_ext_trg6	tim2_trgo
adc3_ext_trg7, adc4_ext_trg7	tim3_oc1
adc3_ext_trg8, adc4_ext_trg8	tim3_oc2 / tim4_oc1(1)
adc3_ext_trg9, adc4_ext_trg9	tim3_trgo / tim4_trgo(1)
adc3_ext_trg10, adc4_ext_trg10	tim6_trgo
adc3_ext_trg11, adc4_ext_trg11	tim8_oc1
adc3_ext_trg12, adc4_ext_trg12	tim8_trgo
adc3_ext_trg13, adc4_ext_trg13	tim8_trgo2
adc3_ext_trg14, adc4_ext_trg14	tim9_trgo
adc3_ext_trg15, adc4_ext_trg15	tim15_trgo
adc3_ext_trg16, adc4_ext_trg16	tim20_oc1
adc3_ext_trg17, adc4_ext_trg17	tim20_trgo
adc3_ext_trg18, adc4_ext_trg18	tim20_trgo2
adc3_ext_trg19, adc4_ext_trg19	lptim1_ch1
adc3_ext_trg20, adc4_ext_trg20	hrtim1_adc_trg1
adc3_ext_trg21, adc4_ext_trg21	hrtim1_adc_trg2
adc3_ext_trg22, adc4_ext_trg22	hrtim1_adc_trg3
adc3_ext_trg23, adc4_ext_trg23	hrtim1_adc_trg4
adc3_ext_trg24, adc4_ext_trg24	hrtim1_adc_trg5
adc3_ext_trg25, adc4_ext_trg25	hrtim1_adc_trg6
adc3_ext_trg26, adc4_ext_trg26	hrtim2_adc_trg1
adc3_ext_trg27, adc4_ext_trg27	hrtim2_adc_trg2

Table 241. ADC1/2/3/4 interconnection (continued)

Signal name	Source/destination
adc3_ext_trg28, adc4_ext_trg28	hrtim2_adc_trg3
adc3_ext_trg29, adc4_ext_trg29	hrtim2_adc_trg4
adc3_ext_trg30, adc4_ext_trg30	hrtim2_adc_trg5
adc3_ext_trg31, adc4_ext_trg31	hrtim2_adc_trg6
adc3_jext_trg0, adc4_jext_trg0	exti3
adc3_jext_trg1, adc4_jext_trg1	tim1_oc3
adc3_jext_trg2, adc4_jext_trg2	tim1_oc4
adc3_jext_trg3, adc4_jext_trg3	tim1_trgo
adc3_jext_trg4, adc4_jext_trg4	tim1_trgo2
adc3_jext_trg5, adc4_jext_trg5	tim2_trgo
adc3_jext_trg6, adc4_jext_trg6	tim3_trgo
adc3_jext_trg7, adc4_jext_trg7	tim7_trgo
adc3_jext_trg8, adc4_jext_trg8	tim4_oc3 / tim8_oc2(1)
adc3_jext_trg9, adc4_jext_trg9	tim4_trgo / tim8_oc4(1)
adc3_jext_trg10, adc4_jext_trg10	tim8_trgo
adc3_jext_trg11, adc4_jext_trg11	tim8_trgo2
adc3_jext_trg12, adc4_jext_trg12	tim12_trgo
adc3_jext_trg13, adc4_jext_trg13	tim15_trgo
adc3_jext_trg14, adc4_jext_trg14	tim16_oc1
adc3_jext_trg15, adc4_jext_trg15	tim17_oc1
adc3_jext_trg16, adc4_jext_trg16	tim20_oc2
adc3_jext_trg17, adc4_jext_trg17	tim20_trgo
adc3_jext_trg18, adc4_jext_trg18	tim20_trgo2
adc3_jext_trg19, adc4_jext_trg19	lptim1_ch2
adc3_jext_trg20, adc4_jext_trg20	hrtim1_adc_trig1
adc3_jext_trg21, adc4_jext_trg21	hrtim1_adc_trig2
adc3_jext_trg22, adc4_jext_trg22	hrtim1_adc_trig3
adc3_jext_trg23, adc4_jext_trg23	hrtim1_adc_trig4
adc3_jext_trg24, adc4_jext_trg24	hrtim1_adc_trig5
adc3_jext_trg25, adc4_jext_trg25	hrtim1_adc_trig6
adc3_jext_trg26, adc4_jext_trg26	hrtim2_adc_trig1
adc3_jext_trg27, adc4_jext_trg27	hrtim2_adc_trig2
adc3_jext_trg28, adc4_jext_trg28	hrtim2_adc_trig3
adc3_jext_trg29, adc4_jext_trg29	hrtim2_adc_trig4

Table 241. ADC1/2/3/4 interconnection (continued)

Signal name	Source/destination
adc3_jext_trg30, adc4_jext_trg30	hrtim2_adc_trig5
adc3_jext_trg31, adc4_jext_trg31	hrtim2_adc_trig6

1. This input can be selected through the option register.

25.4.3 ADC clocks

Dual clock domain architecture

The dual clock-domain architecture means that the ADC kernel clock is independent from the AHB bus clock that is used to access ADC registers.

The adc_ker_ck input clock can be selected between different clock sources (see *Figure 284: ADC clock scheme*). This selection is done in the RCC (refer to section *Reset and clock control (RCC)* for more information):

1. The ADC clock can be provided by an internal or external clock source, which is independent and asynchronous from the AHB clock.
2. The ADC clock can be derived from AHB clock (selected in RCC).

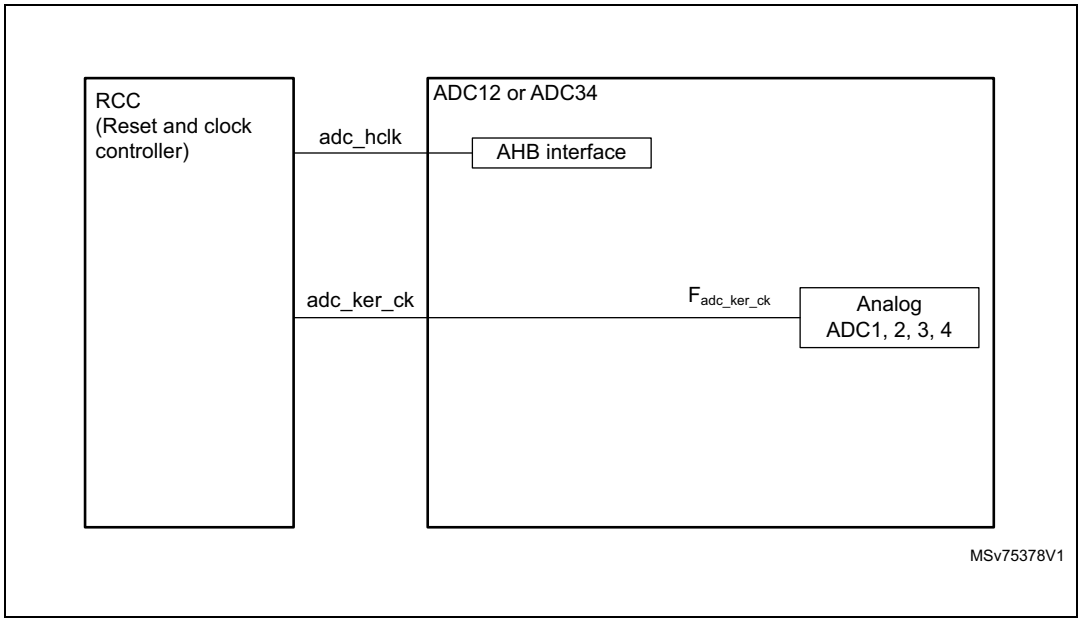
Option **1** has the advantage of achieving the maximum ADC clock frequency whatever the AHB clock scheme selected.

Option **2** corresponds to a pseudosynchronous clock. This can be useful when the ADC is triggered by a timer and the application requires that the ADC is accurately triggered without any uncertainty (otherwise, an uncertainty of the trigger instant time is added by the resynchronizations between the two clock domains). This accurate trigger is supported only by trgo or trgo2 timer triggers.

For further details on the synchronization between ADC and timers, refer to section *ADC synchronization* in each timer section, and to section *Peripheral clock distribution* in the RCC section.

The clock is configured through the RCC and must be compliant with the operating frequency specified in the device datasheet.

Figure 284. ADC clock scheme



Clock ratio constraint between ADC clock and AHB clock

There are no constraints to respect for the ratio between the ADC clock and the AHB clock. However, the ratio must be carefully chosen to avoid any overrun especially if the clock AHB is much slower than the ADC clock.

When adc_hclk operates at a higher frequency than adc_ker_ck, writing to the AHB register does not immediately update the ADC configuration due to clock domain crossing delays. To guarantee that the configuration is properly applied within the adc_ker_ck domain, it is necessary to wait for four adc_ker_ck clock cycles after the AHB register update.

25.4.4 ADC connectivity

ADC inputs are connected to external channels as well as internal sources as described below.

Figure 285. ADC1 connectivity

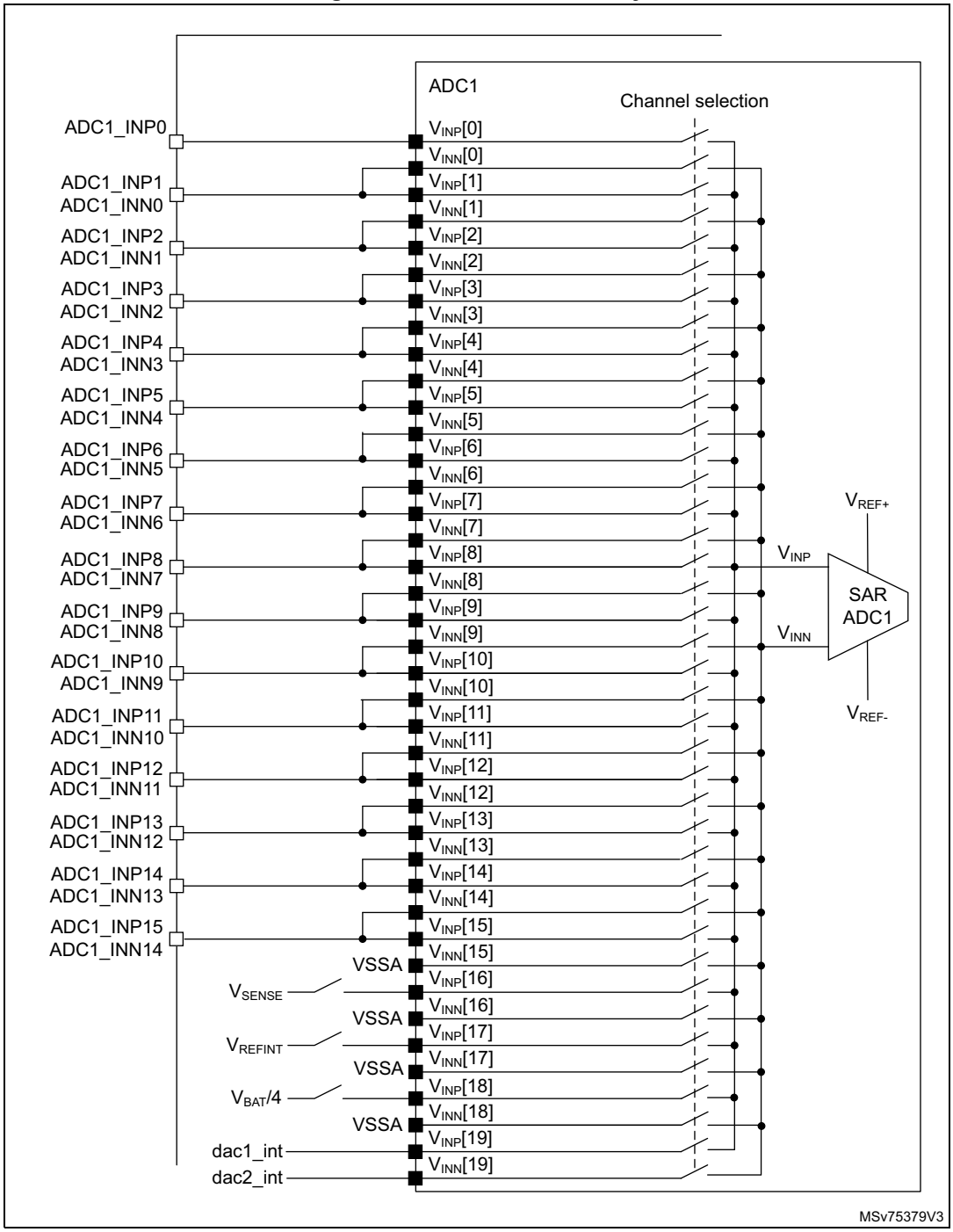


Figure 286. ADC2 connectivity

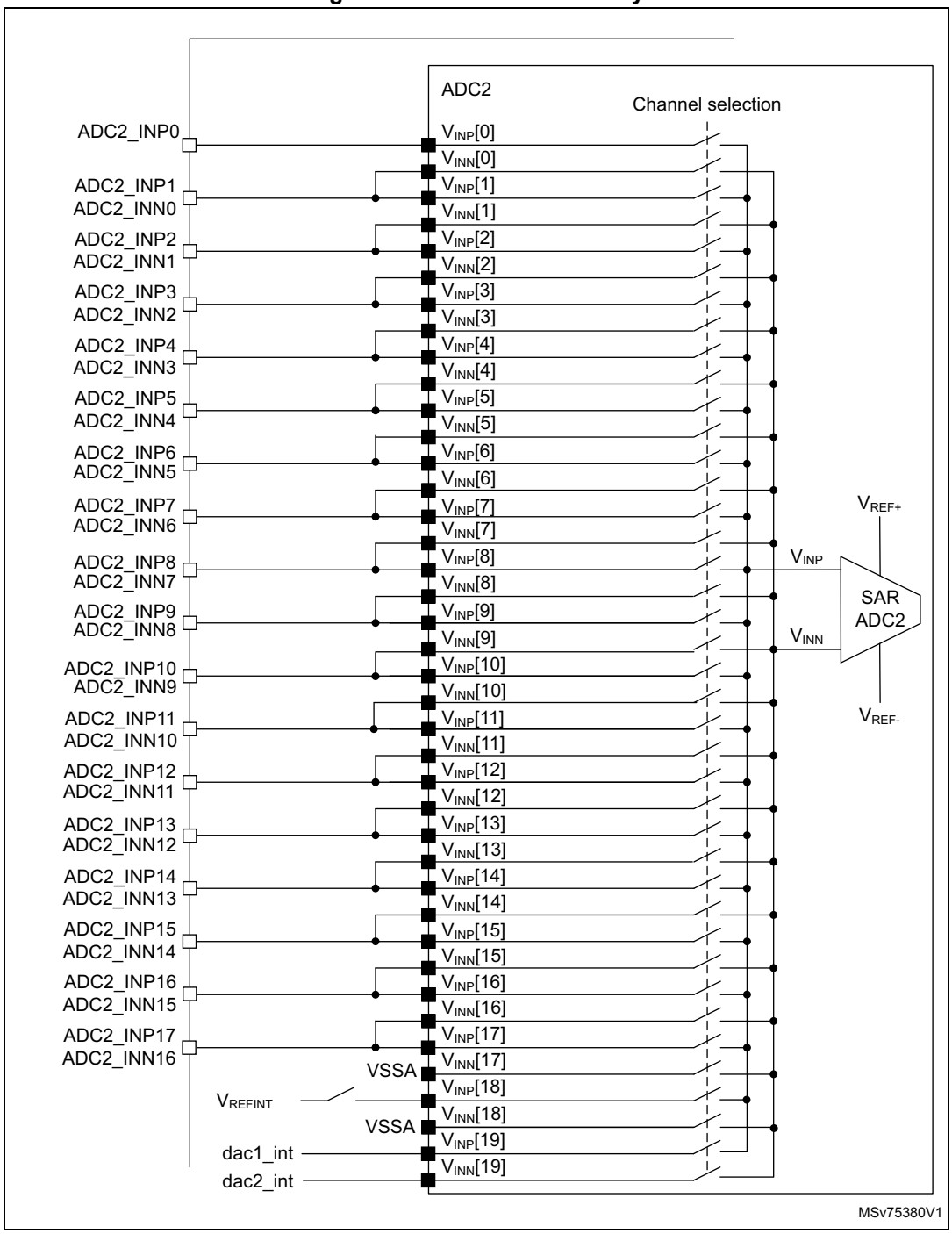


Figure 287. ADC3 connectivity

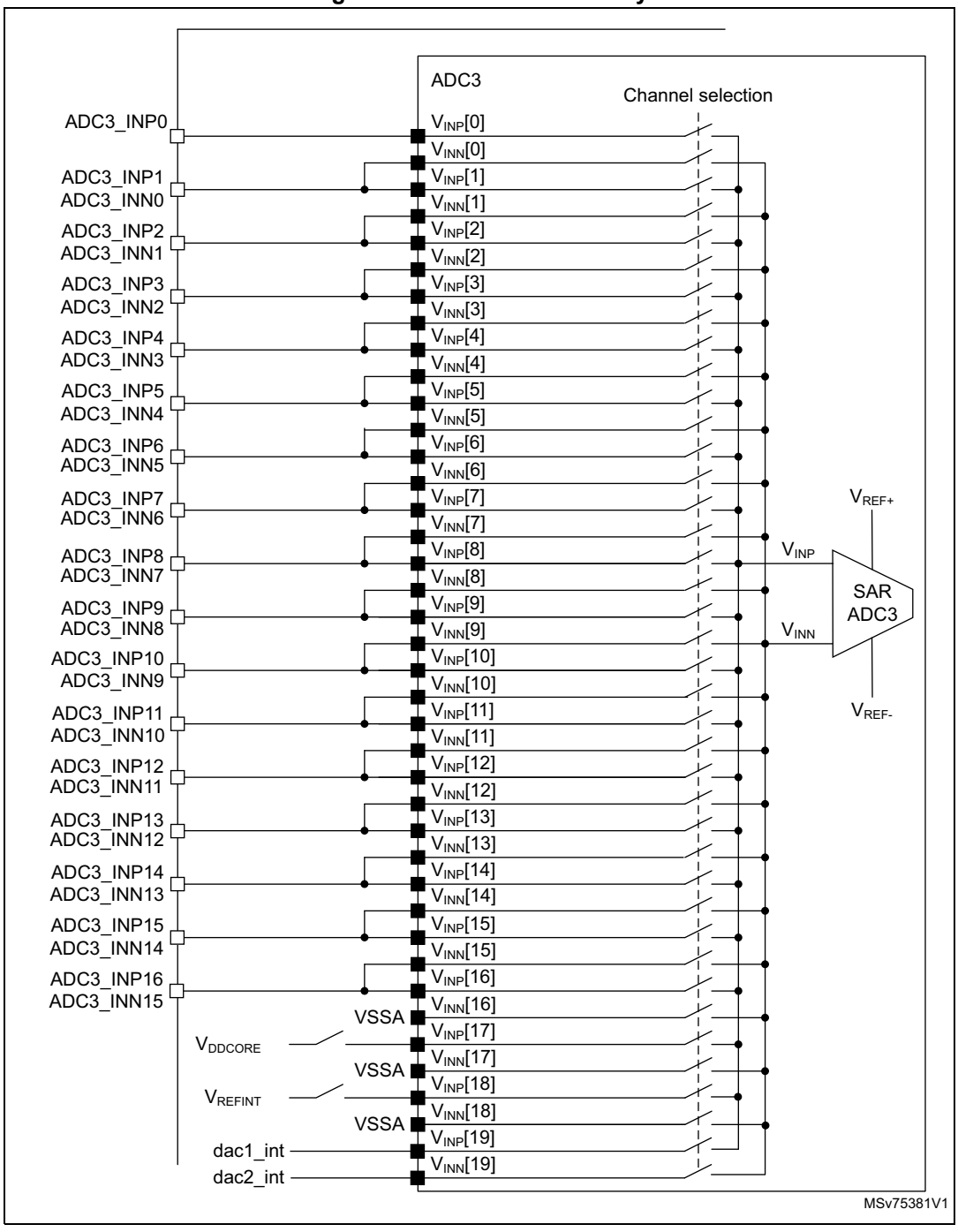
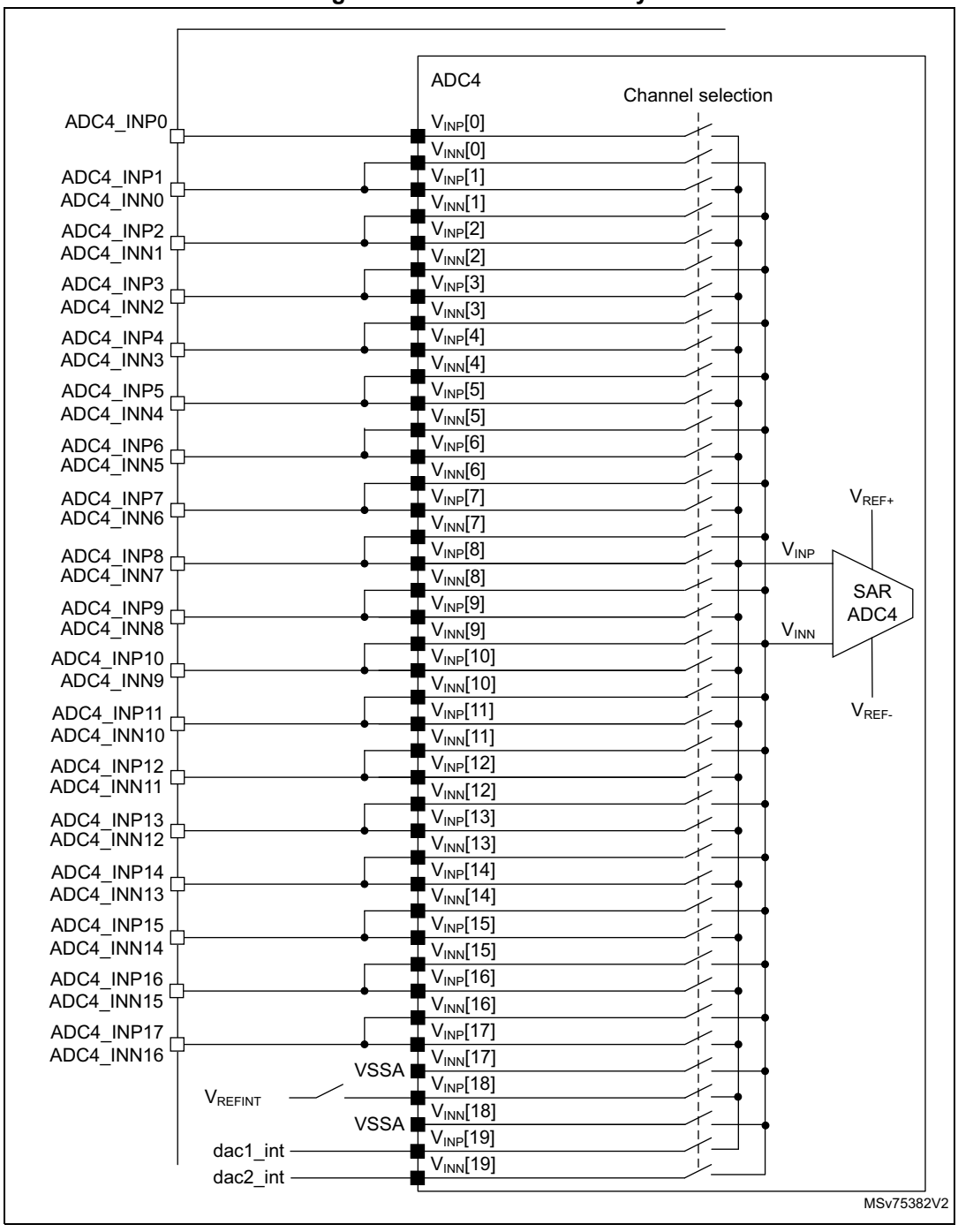


Figure 288. ADC4 connectivity



25.4.5 Slave AHB interface

The ADCs implement an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and it can generate AHB error in case of wrong HSIZE.

25.4.6 ADC Deep-power-down mode (DEEPPWD)

By default, the ADC is in deep-power-down mode where its supply voltage is internally switched off to reduce the leakage currents (the reset state of DEEPPWD bit is 1 in the ADC_CR register).

To activate the ADC, first exit the Deep-power-down mode by clearing the DEEPPWD bit.

When ADC operations are complete, the ADC can be disabled by clearing the ADEN bit of the ADC_CR register.

Power can be saved by entering ADC Deep-power-down mode again (by setting DEEPPWD of ADC_CR register). This is particularly interesting before entering Stop mode.

25.4.7 Single-ended and differential input channels

ADC channels can be configured to be either single-ended inputs or differential inputs by programming DIFSEL[i] bits in the ADC_DIFSEL register. This configuration must be programmed when no ADC conversion is ongoing. Note that the DIFSEL[i] bits corresponding to single-ended channels must be programmed at 0.

In single-ended input mode, the analog voltage to be converted for channel "*i*" is the difference between the external voltage $V_{INP[i]}$ and V_{REF-}.

In differential input mode, the analog voltage to be converted for channel "*i*" is the difference between the external voltage $V_{INP[i]}$ (positive input) and $V_{INN[i]}$ (negative input).

In differential mode, the output data are unsigned data. When $V_{INP[i]}$ equals V_{REF-}, $V_{INN[i]}$ equals V_{REF+} and the output data is 0x000 (12-bit resolution mode). When $V_{INP[i]}$ equals V_{REF+}, $V_{INN[i]}$ equals V_{REF-} and the output data is 0xFFF.

$$\text{Converted value} = \frac{\text{ADC_Full_Scale}}{2} \times \left[1 + \frac{V_{INP} - V_{INN}}{V_{REF+}}\right]$$

When the ADC is configured in differential mode, the input signals are supposed to be differential (the common-mode voltage must be fixed at $V_{REF+}/2$ voltage).

Internal channels are used in single-ended mode only.

For a complete description of how the input channels are connected for each ADC, refer to *Section 25.4.4: ADC connectivity*.

Caution: When configuring channel "*i*" in differential input mode, its negative input voltage $V_{INN[i]}$ is connected to another channel. As a consequence, this channel is no longer usable in single-ended or differential mode, and must never be configured to be converted. Some channels

are shared between ADCs: this can make the channel on the other ADCs unusable. The only exception is interleaved mode for ADC master and the slave.

25.4.8 Pseudodifferential mode

The ADC can operate in pseudodifferential mode instead of differential modes when DIFSELx = 1. In this mode, differential inputs are supported but the negative input $V_{INN[i]}$ is fixed to V_{REF-} or to $V_{REF+}/2$ voltage.

The pseudodifferential mode is selected by setting the PDIFF bit of the ADC_CFGR2 register. This changes the behavior of the differential channel selected via the ADC_DIFSEL register. In addition, the negative input reference voltage can be selected between V_{REF-} and $V_{REF+}/2$ through the PDIFFREF bit of the ADC_CFGR2 register.

When the ADC is configured in pseudodifferential mode:

- When $V_{INP[i]}$ equals 0 V, the output data is 0x000.
- When $V_{INP[i]}$ equals V_{REF+}, the output data is 0xFFF,

Note: The behavior is the same as in single-ended mode except that $V_{INN[i]}$ is fixed to 0 V or $V_{REF+}/2$.

The pseudodifferential mode can be used to reject the common mode noise that exists between the ADC shunt resistor bottom voltage (ground) and top voltage (voltage to be converted by the ADC).

The pseudodifferential mode does not affect single-end channel conversions.

The pseudodifferential and standard differential modes cannot be used simultaneously in the same scan conversion sequence. If pseudodifferential or differential conversion is required, stop and disable the ADC to change the PDIFF setting.

25.4.9 Calibration (ADCAL, ADCALDIF, ADC_CALFACT)

Each ADC provides an automatic calibration procedure that drives all the calibration sequences including the power-on/off sequence of the ADC. During the procedure, the ADC calculates a calibration factor, which is 7-bit wide and is applied internally to the ADC until the next ADC power-off. During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.

Calibration is preliminary to any ADC operation. It removes the offset error which may vary from chip to chip due to process or bandgap variation.

The calibration factor to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Clear ADCALDIF before launching a calibration that is applied for single-ended input conversions.
- Set ADCALDIF before launching a calibration that is applied for differential input conversions.

The calibration is then initiated by software by setting bit ADCAL = 1. Calibration can only be initiated when the ADC is disabled (when ADEN = 0). The ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT_S[6:0] or CALFACT_D[6:0] of the ADC_CALFACT register (depending on single-ended or differential input calibration)

The internal analog calibration is kept if the ADC is disabled (ADEN = 0). However, if the ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before reenabling the ADC.

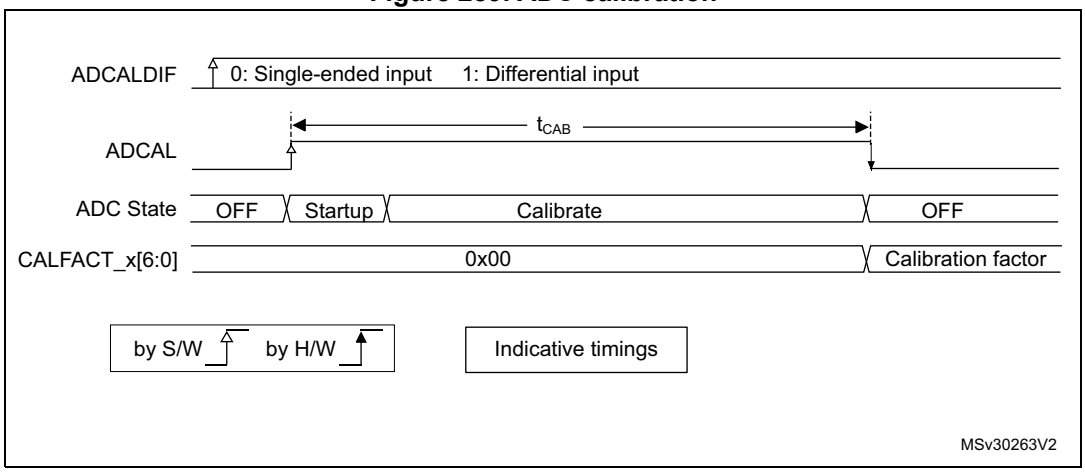
The internal analog calibration is lost each time the power of the ADC is removed (for example, when the product enters Standby). In this case, to avoid spending time recalibrating the ADC, it is possible to rewrite the calibration factor into the ADC_CALFACT register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

The calibration factor can be written if the ADC is enabled but not converting (ADEN = 1 and ADSTART = 0 and JADSTART = 0). Then, at the next start of conversion, the calibration factor is automatically injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion. It is recommended to recalibrate when the V_{REF+} voltage changed more than 10% (see *Section 25.4.37: Monitoring the internal voltage reference*).

Software procedure to calibrate the ADC

1. Ensure DEEPPWD = 0, ADVREGEN = 1 and that ADC voltage regulator startup time has elapsed.
2. Ensure that ADEN = 0.
3. Select the input mode for this calibration by clearing ADCALDIF (single-ended input) or by setting ADCALDIF (differential input).
4. Set ADCAL.
5. Wait until ADCAL = 0.
6. The calibration factor can be read from the ADC_CALFACT register.

Figure 289. ADC calibration

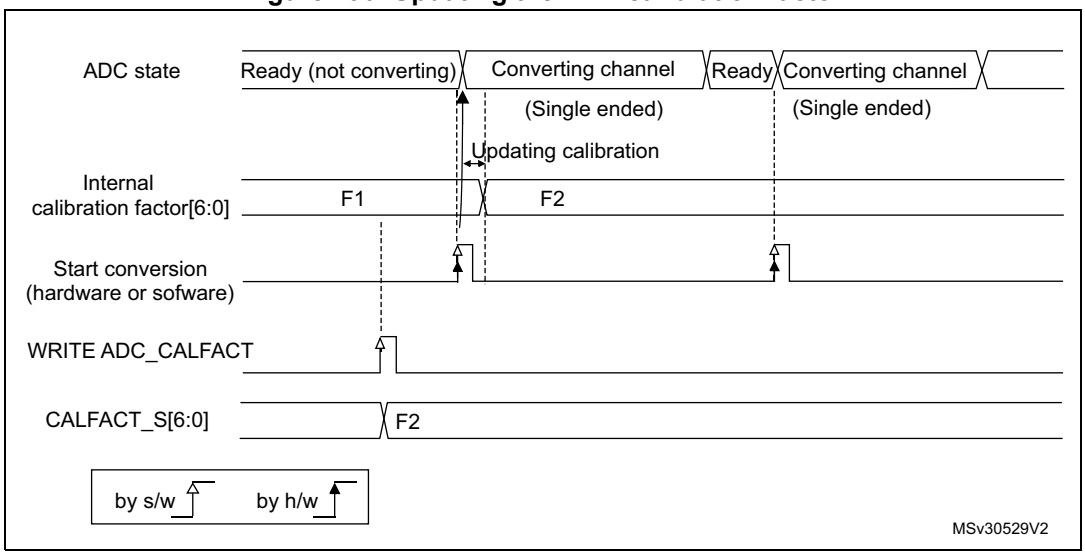


Software procedure to reinject a calibration factor into the ADC

1. Ensure ADSTART = 0 and JADSTART = 0 (no conversion is ongoing).
2. Write CALFACT_S and CALFACT_D with the new calibration factors.
3. When a conversion is launched, the calibration factor is injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits CALFACT_S for single-ended input channel or bits CALFACT_D for differential input channel.

Note: In pseudodifferential mode, only the CALFACT_D calibration factor is used.

Figure 290. Updating the ADC calibration factor

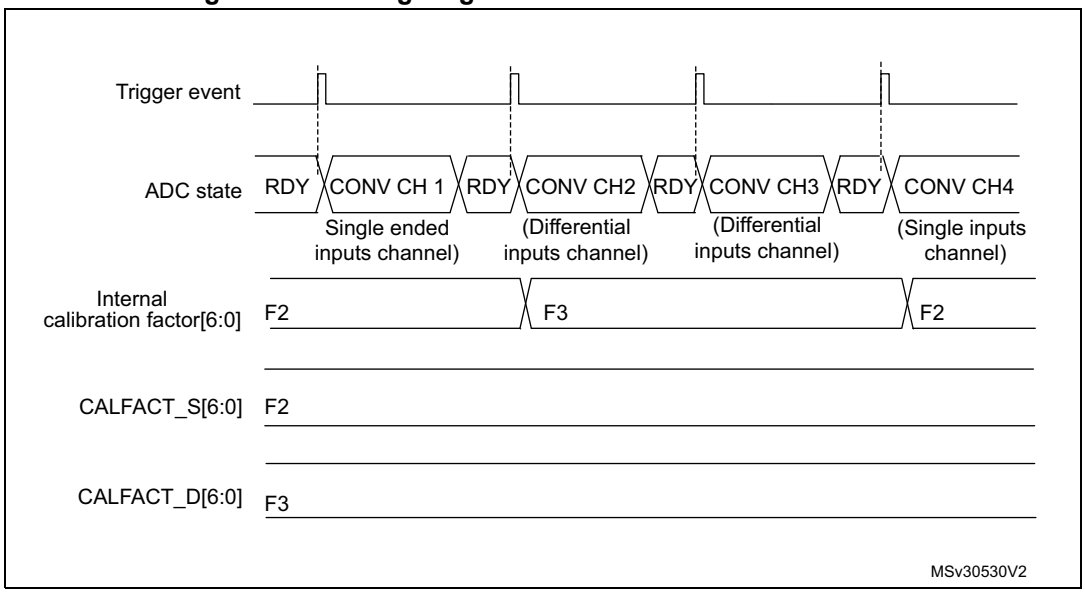


Converting single-ended and differential analog inputs with a single ADC

If the ADC is supposed to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF = 0 and one with ADCALDIF = 1. The procedure is the following:

1. Disable the ADC.
2. Calibrate the ADC in single-ended input mode (with ADCALDIF = 0). This updates the register CALFACT_S[6:0].
3. Calibrate the ADC in differential input modes (with ADCALDIF = 1). This updates the register CALFACT_D[6:0].
4. Enable the ADC, configure the channels and launch the conversions. Each time there is a switch from a single-ended to a differential input channel (and vice versa), the calibration is automatically injected into the analog ADC.

Figure 291. Mixing single-ended and differential channels



Note: The offset calibration in pseudodifferential mode is controlled by the CALFACF_D register.

25.4.10 ADC on-off control (ADEN, ADDIS, ADRDY)

First, follow the procedure described in *Section 25.4.6: ADC Deep-power-down mode (DEEPPWD)*).

Once DEEPPWD is cleared, the ADC can be enabled. It requires a t_{STAB} stabilization time before starting converting accurately (see *Figure 292*).

Two control bits enable or disable the ADC:

- Set ADEN to enable the ADC. The ADRDY flag is set once the ADC is ready for operation.
- Set ADDIS to disable the ADC. ADEN and ADDIS are automatically deasserted by hardware as soon as the analog ADC is effectively disabled.

Regular conversions can then start either by setting ADSTART (refer to *Section 25.4.21: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)*) or when an external trigger event occurs if regular triggers are enabled.

Injected conversions start by setting JADSTART or when an external injected trigger event occurs, if injected triggers are enabled.

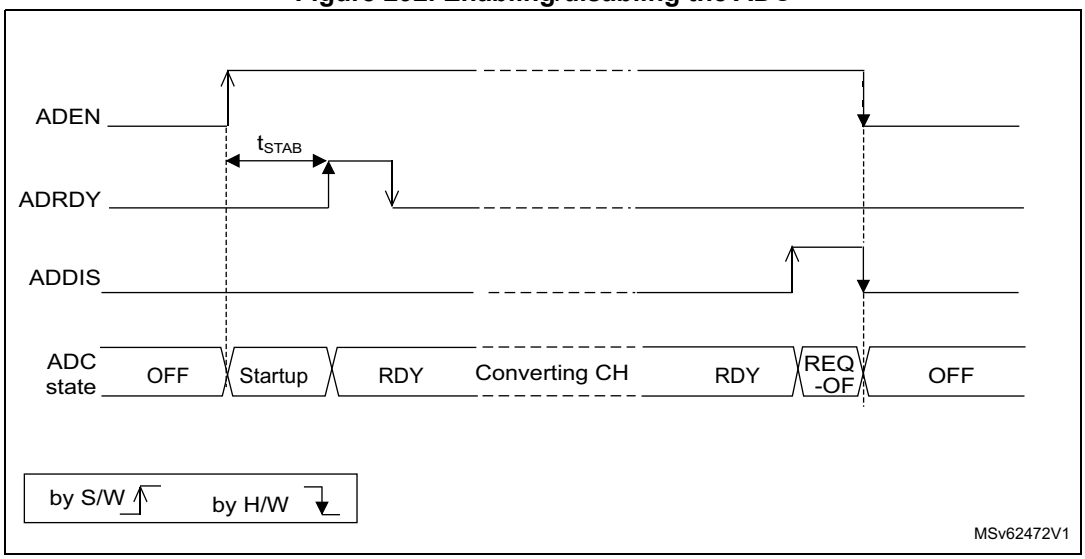
Software procedure to enable the ADC

1. Clear the ADRDY bit in the ADC_ISR register by programming it to 1.
2. Set ADEN.
3. Wait until ADRDY = 1 (ADRDY is set after the ADC startup time). This can be done by using the associated interrupt (ADRDYIE must be set).
4. Clear the ADRDY bit in the ADC_ISR register by programming it to 1 (optional).

Software procedure to disable the ADC

1. Check that both ADSTART = 0 and JADSTART = 0 to ensure that no conversion is ongoing. If required, stop all regular and injected ongoing conversions by setting ADSTP and JADSTP. Then wait until ADSTP = 0 and JADSTP = 0.
2. Set ADDIS.
3. If required by the application, wait for ADEN = 0, until the analog ADC is effectively disabled (ADDIS is automatically reset once ADEN = 0).

Figure 292. Enabling/disabling the ADC



25.4.11 Constraints when writing the ADC control bits

The software is allowed to write the RCC control bits to configure and enable the ADC clock (refer to RCC section) and configure the ADEN bit in the ADC_CR register, only if the ADC is disabled (ADEN must be cleared).

The software can modify the PDIFF and PDIFFREF of the ADC_CFGR2 only when ADEN is cleared.

The software can write the ADSTART, JADSTART and ADDIS control bits in the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

The following constraints apply to all the other control bits of the ADC_CFGRx, ADC_SMPRx, ADC_TRy, ADC_SQRy, ADC_JDRy, ADC_OFRy, ADC_OFCHRy and ADC_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if no regular conversions are ongoing (ADSTART must be cleared).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if no injected conversions are ongoing (JADSTART must be cleared).
- ADC_TRy register thresholds can be modified when an analog-to-digital conversion is ongoing (refer to *Section 25.4.31: Analog window watchdog (AWD1EN, JAWD1EN,*

AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT) for details).

- To modify the all above control bits when the software trigger function is used, an ADSTP or JADSTP must be issued to make sure all activities are stopped, even if ADSTART and JADSTART are cleared.

The software is allowed to write the ADSTP or JADSTP control bits in the ADC_CR register only if the ADC is enabled and if there is no pending request to disable the ADC. In addition, the software can write the ADC_JSQR register, when injected queue are disabled by JQDIS = 1, only if JADSTART is cleared and no injected conversion is ongoing. Refer to *Section 25.6.14: ADC injected sequence register (ADC_JSQR)* for additional details.

Note: Not all forbidden write accesses to ADC control bits are protected by hardware. In some cases, the ADC state may become unknown. To recover from this situation, the ADC must be disabled (ADEN = 0 as well as all the bits of the ADC_CR register).

25.4.12 Channel selection (ADC_SQRy, ADC_JSQR)

The ADC features up to 20 multiplexed channels per ADC, out of which:

- Up to 18 analog inputs coming from GPIO pads (ADCx_INN/INP[i]). Depending on the device, not all of them are available on GPIO pads.
- The ADC is connected to up to 6 internal analog inputs:
  - internal reference voltage (V_{REFINT})
  - Internal temperature sensor (V_{SENSE})
  - V_{BAT} monitoring channel ($V_{BAT}/4$)
  - V_{DDCORE} voltage
  - DAC internal channel

  To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming VREFEN, TSEN, VBATEN in the ADCC_CCR registers, or by enabling the corresponding bit in the *ADC option register (ADC_OR)*.

Refer to *ADC interconnection* tables in *Section 25.4.2: ADC pins and internal signals* for the connection of the above internal analog inputs to ADC pins.

The conversions can be organized in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADCx_INN/INP3, ADCx_INN/INP8, ADCx_INN/INP2, ADCx_INN/INP2, ADCx_INN/INP0, ADCx_INN/INP2, and ADCx_INN/INP2.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRy registers. The total number of conversions in the regular group must be written in the LEN[3:0] bits in the ADC_SQR1 register.
- An **injected group** is composed of up to four conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the JLEN[1:0] bits in the ADC_JSQR register.

ADC_SQRy registers must not be modified while a regular conversion is ongoing. ADC regular conversions must consequently be stopped by setting ADSTP (refer to *Section 25.4.20: Stopping an ongoing conversion (ADSTP, JADSTP)*).

In addition, the ADC_JSQR register must not be modified while an injected conversion is ongoing. ADC injected conversions must consequently be stopped by setting JADSTP, except when JQDIS is cleared.

25.4.13 Channel preselection register (ADC_PCSEL)

The PCSEL bit of the ADC_PCSEL register controls the analog switch integrated in the I/O.

For each channel selected through SQRx or JSQRx bits, the corresponding PCSEL bit must be configured in advance in the ADC_PCSEL register. The ADC input multiplexer selects the ADC input according to SQRx and JSQRx configuration with very high speed. The analog switch integrated in the I/O cannot react as fast as the ADC multiplexer. To avoid the delay due to the analog switch control on the I/O, it is necessary to preselect the input channels that are selected through the SQRx and JSQRx. The selection is based on the V_{INP} of each ADC input. For example, if the ADC converts ADCx_INN/INP1, PCSEL1 bit must also be set in ADC_PCSEL register. If ADCx_INN/INP1 is configured as a differential channel, both PCSEL1 and PCSEL0 must be set, which corresponds to ADC1_INN1 and ADC2_INN1.

Note: Configuring the PCSEL bit is not necessary for the internal channels (such as V_{REFINT}).

25.4.14 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be sufficient for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time, which is programmable through the SMP[2:0] bits of the ADC_SMPR1 and ADC_SMPR2 registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 2.5 ADC clock cycles
- SMP = 001: 4.5 ADC clock cycles
- SMP = 010: 7.5 ADC clock cycles
- SMP = 011: 12.5 ADC clock cycles
- SMP = 100: 24.5 ADC clock cycles
- SMP = 101: 47.5 ADC clock cycles
- SMP = 110: 256.5 ADC clock cycles
- SMP = 111: 848.5 ADC clock cycles

The total conversion time is calculated as follows (12-bit mode):

$$T_{CONV} = \text{Sampling time} + 13 \text{ ADC clock cycles}$$

Example

With $F_{adc_ker_ck}$ = 32 MHz and a sampling time of 3 ADC clock cycles:

$$T_{CONV} = (3 + 13) \times \text{ADC clock cycles} = 16 \text{ ADC clock cycles} = 500 \text{ ns}$$

The ADC notifies the end of the sampling phase by asserting the EOSMP flag (only for regular conversion).

Constraints on the sampling time

For each channel, SMP[2:0] bits must be programmed to respect a minimum and a maximum sampling times as specified in the ADC characteristics section of the datasheets.

Bulb sampling mode

The bulb sampling mode enables to obtain longer sampling time.

When the BULB bit is set in the ADC_CFGR2 register, the sampling period starts immediately after the last ADC conversion. A hardware or software trigger starts the conversion after the sampling time has been programmed in ADC_SMPRx registers. The very first ADC conversion after the ADC is enabled, is performed with the sampling time programmed in SMP bits. The bulb mode is effective starting from the second conversion.

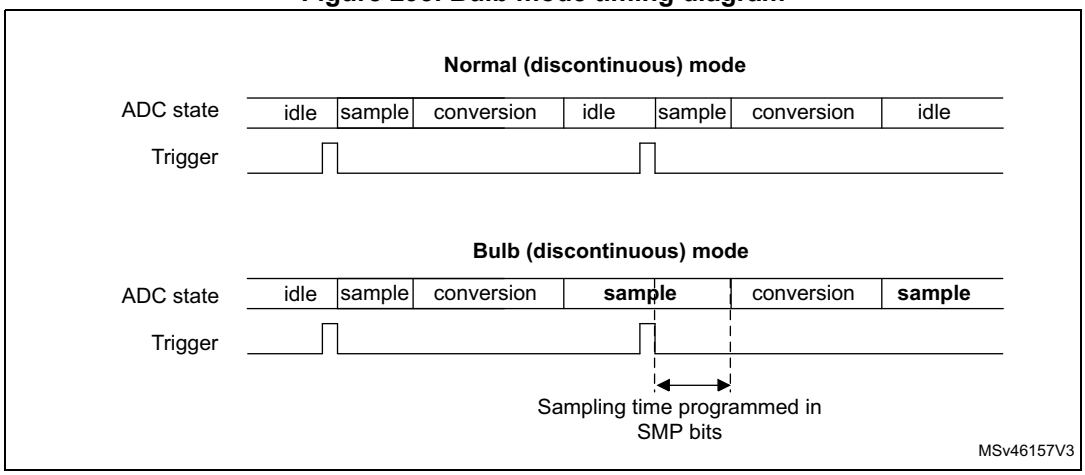
The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

Bulb mode is supported exclusively in regular conversion; dual mode is not supported.

The BULB bit can be modified only when the ADEN bit of the ADC_CR register is cleared.

When the BULB bit is set, it is not allowed to set the SMPTRIG bit in ADC_CFGR2.

Figure 293. Bulb mode timing diagram



Sampling time control trigger mode

When the SMPTRIG bit is set, the sampling time programmed through SMPx bits is not applicable. The sampling time is controlled by the trigger signal edge.

When a hardware trigger is selected (EXTEN[1:0] = 01), each rising edge of the trigger signal starts the sampling period. A falling edge ends the sampling period and starts the conversion.

Due to the synchronization mechanism, the minimum allowed pulse width is seven adc_ker_ck periods.

When a software trigger is selected (EXTEN[1:0] = 00), the software trigger is not the ADSTART bit of ADC_CR but the SWTRIG bit. The SWTRIG bit has to be set to start the sampling period, and it has to be cleared to end the sampling period and start the conversion. In this mode, the minimum sampling time is limited to seven ADC clock cycles.

The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

This mode is only compatible with the discontinuous mode. DISCNUM must be cleared for regular conversion mode. It is not supported for injected conversion mode and autoinjection mode.

When the SMPTRIG bit is set, setting the BULB bit is not allowed and only regular conversion is supported.

The sampling time control trigger mode is not compatible with the following dual modes: DUAL[4:0] = 0b00010, 0b00011, 0b00101, 0b00110, 0b00111, and 0b01001.

I/O analog switch voltage booster

The resistance of the I/O analog switches increases when V_{DDA} voltage is too low. The sampling time must consequently be adapted accordingly (refer to the device datasheet for the corresponding electrical characteristics). This resistance can be minimized at low V_{DDA} voltage by enabling an internal voltage booster through the BOOSTEN or BOOSTVDDSEL bit of the SBS_PMCR register.

25.4.15 Single conversion mode

To enable the single conversion mode for regular channels, clear both CONT and DISCEN bits in ADC_CFGR1 register. To enable it on injected channel, clear JDISCEN bit.

In single conversion mode, the ADC performs once all the regular conversions of the programmed channels. This mode is started by one of the following events:

- ADSTART bit set in the ADC_CR register (for a regular channel)
- JADSTART bit set in the ADC_CR register (for an injected channel)
- An external hardware trigger event (for a regular or injected channel)

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 32-bit 8-level FIFO accessible through the ADC_DR register,
- The EOC (end of regular conversion) flag is set, and
- An interrupt is generated if the EOCIE bit is set.

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 32-bit ADC_JDRy registers,
- The JEOC (end of injected conversion) flag is set, and
- An interrupt is generated if the JEOCIE bit is set.

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set, and
- An interrupt is generated if the EOSIE bit is set.

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set, and
- An interrupt is generated if the JEOSIE bit is set.

The ADC then stops until a new external regular or injected trigger occurs or until the ADSTART or JADSTART bit is set again.

Note: To convert a single channel, program a sequence with a length of 1.

25.4.16 Continuous conversion mode (CONT = 1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically restarts and continuously converts all the conversions of the sequence. This mode is started by setting the CONT bit of the ADC_CFGR1 register, either by an external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the eight 32-bit word FIFO accessible through the ADC_DR register.
- The EOC (end of conversion) flag is set, and
- An interrupt is generated if the EOCIE bit is set.

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set, and
- An interrupt is generated if the EOSIE bit is set.

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

It is not possible to have both discontinuous mode and continuous mode enabled, this means that it is forbidden to set both DISCEN and CONT bits.

Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit) (refer to Section : Autoinjection mode).

25.4.17 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN and CONT = 0 bit in the ADC_CFGR1 register.

It is used to convert a short sequence (subgroup) of n conversions (n ≤ 8) that is part of the sequence of conversions selected in the ADC_SQRy registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR1 register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRy registers until all the conversions in the sequence are done. The total sequence length is defined by the LEN[3:0] bits in the ADC_SQR1 register.

Example:

- DISCEN = 1, n = 3, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - First trigger: channels converted are 1, 2, 3 (an EOC event is generated at the end of each conversion).
  - Second trigger: channels converted are 6, 7, 8 (an EOC event is generated at the end of each conversion).
  - Third trigger: channels converted are 9, 10, 11 (an EOC event is generated at the end of each conversion) and an EOS event is generated after the conversion of channel 11.

  - Fourth trigger: channels converted are 1, 2, 3 (an EOC event is generated at the end of each conversion).
  - ...
- DISCEN = 0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10,11
  - First trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - All the next trigger events relaunch the complete sequence.

Note: The channel numbers referred to in the above example might not be available on all devices.

When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the fourth trigger reconverts the channels 1, 2 and 3 in the first subgroup.

It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN = 1, CONT = 1), the ADC behaves as if continuous mode was disabled.

Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADC_CFGR1 register. It converts the sequence selected in the ADC_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels with DISCNUM = 0.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JLEN[1:0] bits in the ADC_JSQR register.

Example:

- JDISCEN = 1, channels to be converted = 1, 2, 3
  - First trigger: channel 1 converted (a JEOC event is generated)
  - Second trigger: channel 2 converted (a JEOC event is generated)
  - Third trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

Note: The channel numbers referred to in the above example might not be available on all devices.

When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the fourth trigger reconverts the first injected channel 1.

It is not possible to use both autoinjection mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

25.4.18 Starting conversions (ADSTART, JADSTART)

ADC regular conversions can be started by setting ADSTART.

When ADSTART is set, the conversion starts:

- Immediately if EXTEN[1:0] = 00 (software trigger), or
- At the next active edge of the selected regular hardware trigger, if EXTEN[1:0] is not equal to 00.

The application software starts ADC injected conversions by setting JADSTART.

When JADSTART is set, the conversion starts:

- Immediately if JEXTEN[1:0] = 00 (software trigger), or
- At the next active edge of the selected injected hardware trigger, if JEXTEN[1:0] is not equal to 00.

Note: In autoinjection mode (JAUTO = 1), use the ADSTART bit to start regular conversions followed by autoinjected conversions (JADSTART must be kept cleared).

When the hardware trigger is enabled, ADSTART and JADSTART also indicate whether an ADC operation is ongoing. The ADC can be reconfigured while ADSTART and JADSTART are both cleared, indicating that the ADC is idle.

In the case of software triggering, ADSTP and/or JADSTP must be set before ADC reconfiguration.

ADSTART is deasserted by hardware:

- In single mode with software regular trigger (CONT = 0, EXTEN = 0x0): at any end of regular conversion sequence (EOS assertion)
- In discontinuous mode (CONT = 0, DISCEN = 1, DISCNUM= x): at any of subgroup sequence
- In all cases (CONT and EXTEN don’t care), after execution of the ADSTP procedure asserted by the software

Note: In continuous mode (CONT = 1), ADSTART is not deasserted by hardware when EOS is asserted because the sequence is automatically relaunched.

When a hardware trigger is selected in single mode (CONT = 0 and EXTEN ≠ 0x00), ADSTART is not deasserted by hardware when EOS is asserted to help the software that does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.

JADSTART is deasserted by hardware:

- In single mode with software injected trigger (JEXTEN = 0x0): at the end of each injected conversion sequence (JEOS assertion)
- In discontinuous mode (JDISCEN = 1): at the end of any conversion
- In all cases (JEXTEN=x), after execution of the JADSTP procedure asserted by the software

Note: When the software trigger is selected, the ADSTART bit must not be set if the EOC flag is still high.

In bulb mode (BULB = 1), the previous end-of-conversion automatically starts the sampling, then setting ADSTART starts the conversion including the programmed sampling time.

In sampling time control trigger mode (SMPTRIG = 1), SWTRIG bit must be set to start the conversion.

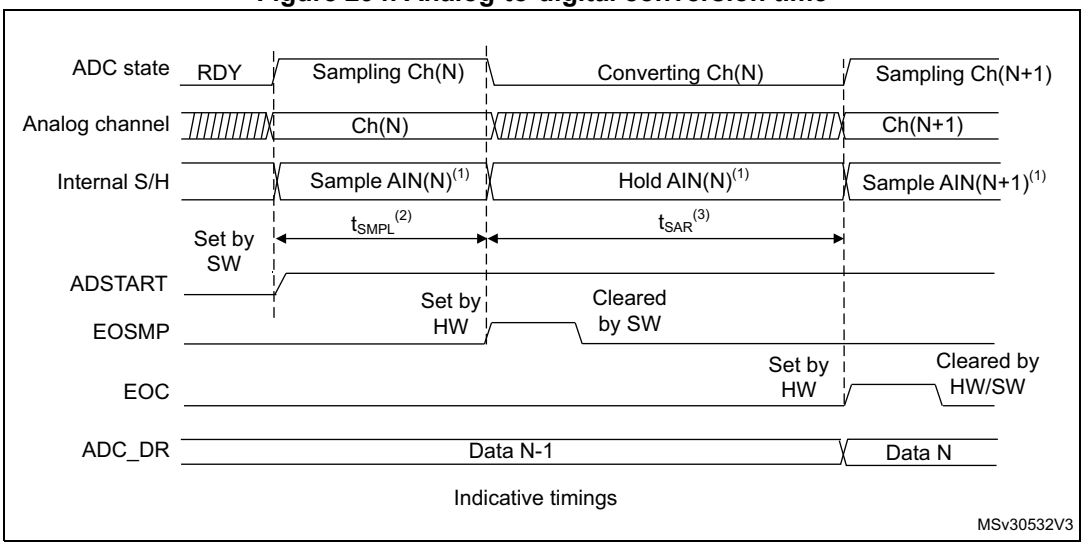
25.4.19 Timing

The elapsed time between the start of a conversion and the end of a conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$T_{CONV} = T_{SMPL} + T_{SAR} = [3\ _{|min} + 13\ _{|12bit}] \times T_{ADC_CLK}$

$T_{CONV} = T_{SMPL} + T_{SAR} = 93.75\ ns\ _{|min} + 406.25\ ns\ _{|12bit} = 500\ ns$ (for $F_{adc_ker_ck} = 32$ MHz)

Figure 294. Analog-to-digital conversion time



1. AIN represents the input voltage captured by the internal Sample and Hold (S/H) capacitor.
2. t_{SMPL} depends on SMP[2:0].
3. t_{SAR} depends on RES[2:0].

Note: The bulb and the sampling time control trigger modes are not described in Figure 294: Analog-to-digital conversion time.

25.4.20 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop ongoing regular conversions by setting ADSTP and ongoing injected conversions by setting JADSTP.

Stopping conversions resets the ongoing ADC operation. The ADC can then be reconfigured (for example by changing the channel selection or the trigger). It is then ready for a new operation.

Injected conversions can be stopped while regular conversions are still ongoing and vice versa. This allows, for instance, the reconfiguration of the injected conversion sequence and triggers while regular conversions are still ongoing (and vice versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with the partial result discarded (the ADC_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (the ADC_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC restarts a new sequence).

Once this procedure is complete, ADSTP/ADSTART bits (for regular conversion), or JADSTP/JADSTART bits (for injected conversion) are deasserted by hardware and the software must poll ADSTART (or JADSTART) until the bit is reset before assuming the ADC is completely stopped.

Note: In autoinjection mode (JAUTO = 1), setting the ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).

Figure 295. Stopping ongoing regular conversions

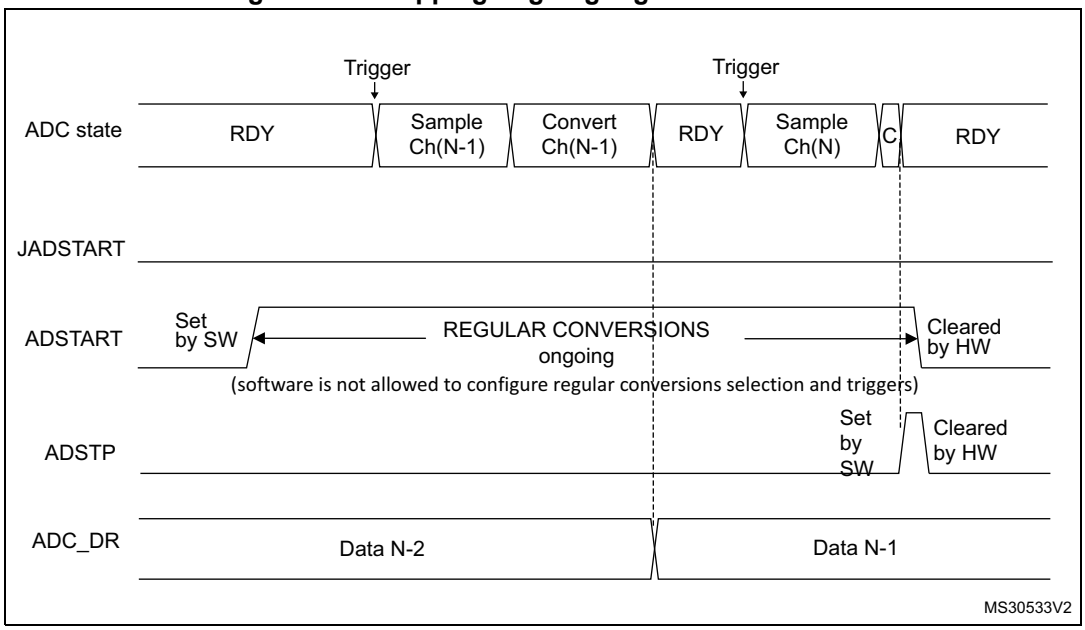
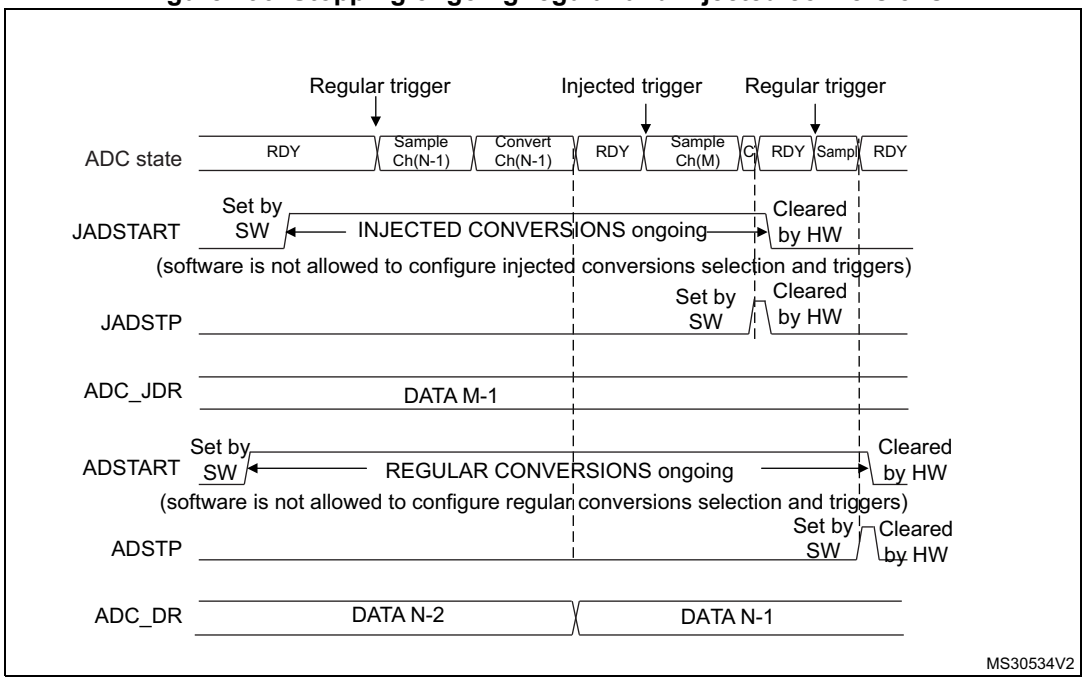


Figure 296. Stopping ongoing regular and injected conversions



25.4.21 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (for example timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events are able to trigger a conversion with the selected polarity.

The regular trigger selection is effective once the software has set ADSTART while the injected trigger selection is effective once the software has set JADSTART bit.

All hardware triggers that occur while a conversion is ongoing are ignored:

- If ADSTART = 0, regular hardware triggers are ignored.
- If JADSTART = 0, injected hardware triggers are ignored.

Table 242 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 242. Configuring the trigger polarity for regular external triggers

EXTEN[1:0]	Source
00	Hardware trigger detection disabled, software trigger detection enabled
01	Hardware trigger with detection on the rising edge (sampling time controlled in SMPTRIG mode)
10	Hardware trigger with detection on the falling edge
11	Hardware trigger with detection on both the rising and falling edges

Note: The polarity of the regular trigger cannot be changed on-the-fly.

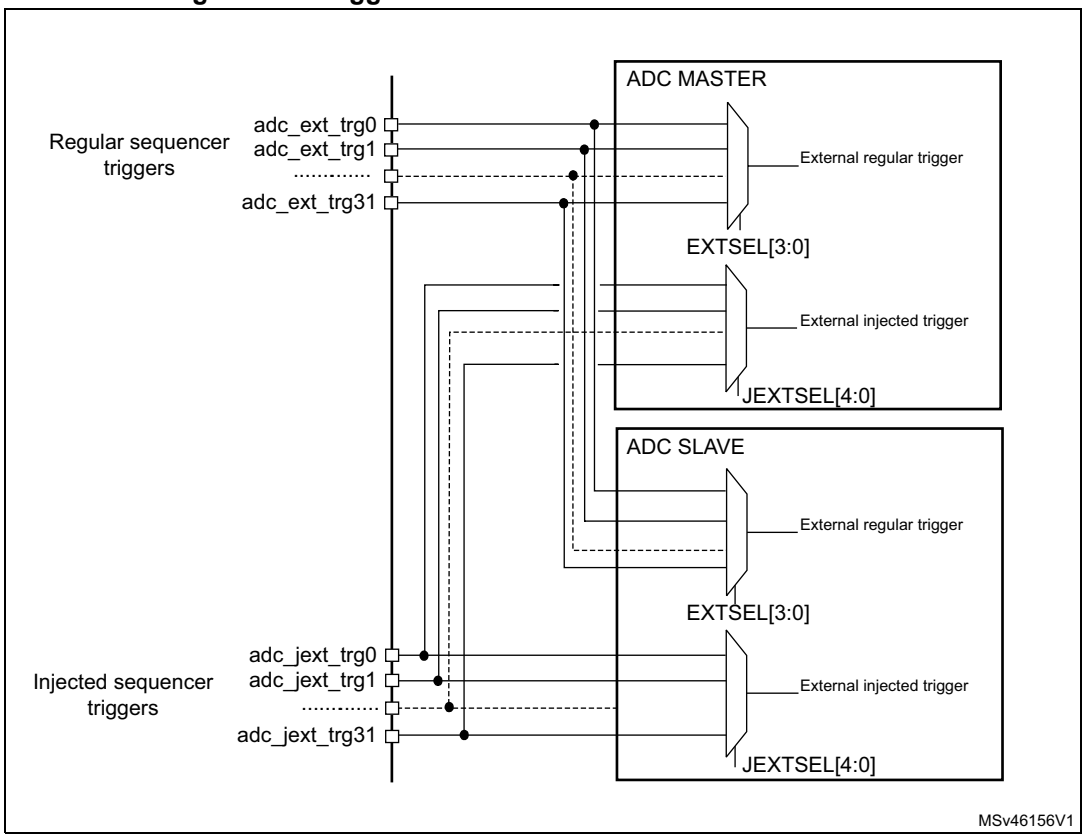
Table 243. Configuring the trigger polarity for injected external triggers

JEXTEN[1:0]	Source
00	Hardware Trigger detection disabled, software trigger detection enabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

The EXTSEL and JEXTSEL control bits select which, out of 32 possible events, can trigger the conversion of the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

Figure 297. Triggers shared between ADC master and slave



Refer to *ADC interconnection* tables in *Section 25.4.2: ADC pins and internal signals* for the list of the external triggers.

25.4.22 Injected channel management

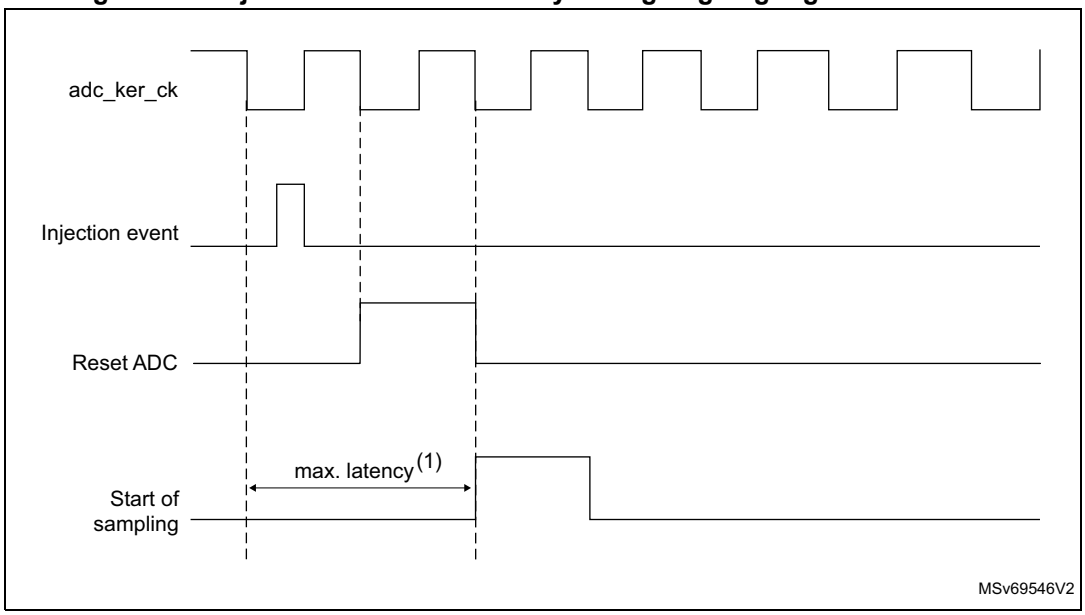
Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC_CFGR1 register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC_CR register.
2. If an external injected trigger occurs, or if the JADSTART bit in the ADC_CR register is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches are launched (all the injected channels are converted once if JDISCEN = 0).
3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence. *Figure 298* shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence.

Figure 298. Injected conversion latency during ongoing regular conversion



1. The maximum latency value can be found in the electrical characteristics of the device datasheet.

Autoinjection mode

If the JAUTO bit in the ADC_CFGR1 register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRy and ADC_JSQR registers.

In this mode, the ADSTART bit in the ADC_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

In this mode, external trigger on injected channels must be disabled.

Note: Autoinjection mode is compatible with discontinuous mode.

25.4.23 Queue of context for injected conversions

A queue of context is implemented to anticipate up to two contexts for the next injected sequence of conversions. JQDIS bit of the ADC_CFGR1 register must be reset to enable this feature. Only hardware-triggered conversions are possible when the context queue is enabled. The queue of context is not supported in dual modes or in auto delay mode.

This context consists of:

- Configuration of the injected triggers (bits JEXTEN[1:0] and JEXTSEL bits in the ADC_JSQR register)
- Definition of the injected sequence (bits JSQx[4:0] and JLEN[1:0] in the ADC_JSQR register)

All the parameters of the context are defined in a single register ADC_JSQR and this register implements a queue of two buffers, allowing the bufferization of up to two sets of parameters:

- The ADC_JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written to the ADC_JSQR register is stored into the queue of context.
- At the beginning, the queue is empty and the first write access to the ADC_JSQR register defines the first context and the ADC is ready to receive injected triggers if JADSTART = 1.
- Once an injected sequence is complete, the queue is consumed and the context changes according to the next ADC_JSQR parameters stored in the queue. This new context is applied for the next injected sequence of conversions.
- A queue overflow occurs when writing to the ADC_JSQR register while the queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of ADC_JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two behaviors are possible when the queue becomes empty, depending on the value of the JQM control bit of the ADC_CFGR1 register:
  - If JQM = 0, the queue is empty just after enabling the ADC, but then it can never be empty during run operations: the queue always maintains the last active.
  - If JQM = 1, the queue becomes empty when end of injected sequence of last context occurs or if the queue is flushed by JADSTP = 1. When this occurs, there is no more context in the queue and software needs to write new injected context in the ADC_JSQR register.
- Reading the ADC_JSQR register returns the current the ADC_JSQR context which is active at that moment. If JQM = 0, when the ADC_JSQR context is empty, the JSQi is read as 0x00.
- The queue is flushed when stopping injected conversions by setting JADSTP or when disabling the ADC by setting ADDIS:
  - If JQM = 0, the queue is flushed, but the last active context is maintained (ADC_JSQR register kept last value).
  - If JQM = 1, the queue becomes empty.

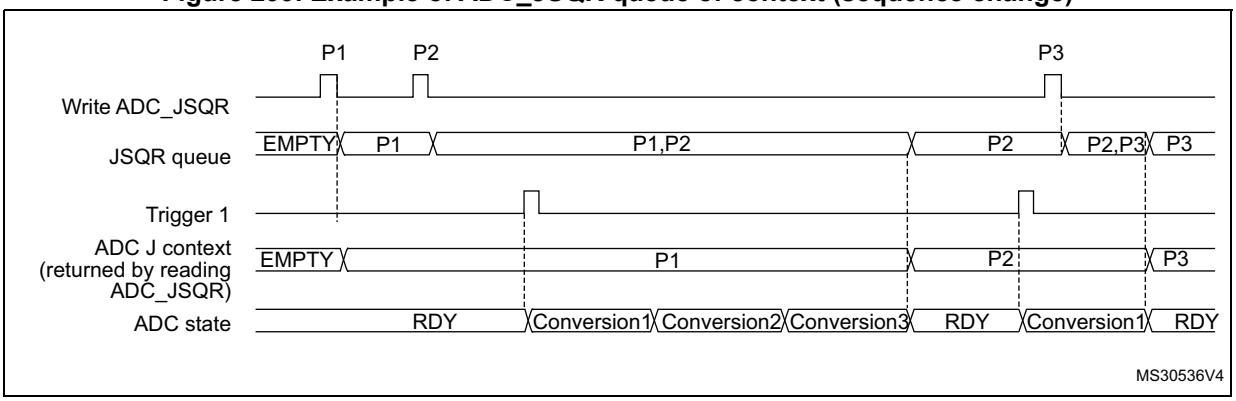
Note: When discontinuous mode (bit JDISCEN = 1) is configured, only JEOS, which marks the end of injected sequence, triggers the transfer of the queue to the active context (JSQR register). See example below (length = 3 for both contexts):

- *First trigger, discontinuous. Sequence 1: context 1, first conversion carried out.*
- *Second trigger, discontinuous. Sequence 1: context 1, second conversion carried out.*
- *Third trigger, discontinuous. Sequence 1: context 1, third conversion carried out. Then context change occurs to context 2.*
- *Fourth trigger, discontinuous. Sequence 2: context 2, first conversion carried out.*
- *Fifth trigger, discontinuous. Sequence 2: context 2, second conversion carried out.*
- *Sixth trigger, discontinuous. Sequence 2: context 2, third conversion carried out. Then context change occurs to next context.*

Behavior when changing the trigger or sequence context

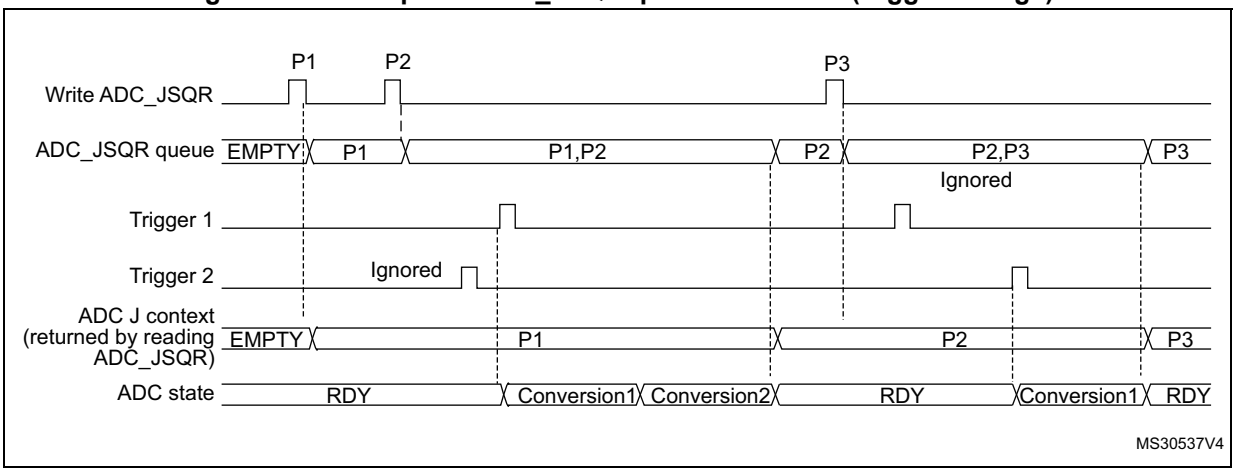
Figure 299 and *Figure 300* show the behavior of the context queue when changing the sequence or the triggers.

Figure 299. Example of ADC_JSQR queue of context (sequence change)



1. Parameters:
   P1: sequence of 3 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 4 conversions, hardware trigger 1

Figure 300. Example of ADC_JSQR queue of context (trigger change)

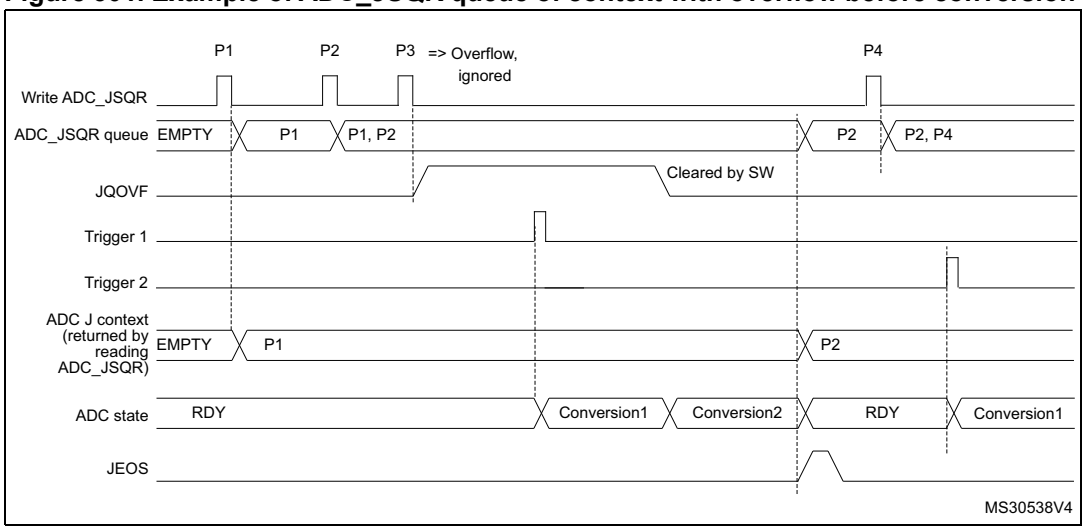


1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 4 conversions, hardware trigger 1

Queue of context: Behavior when a queue overflow occurs

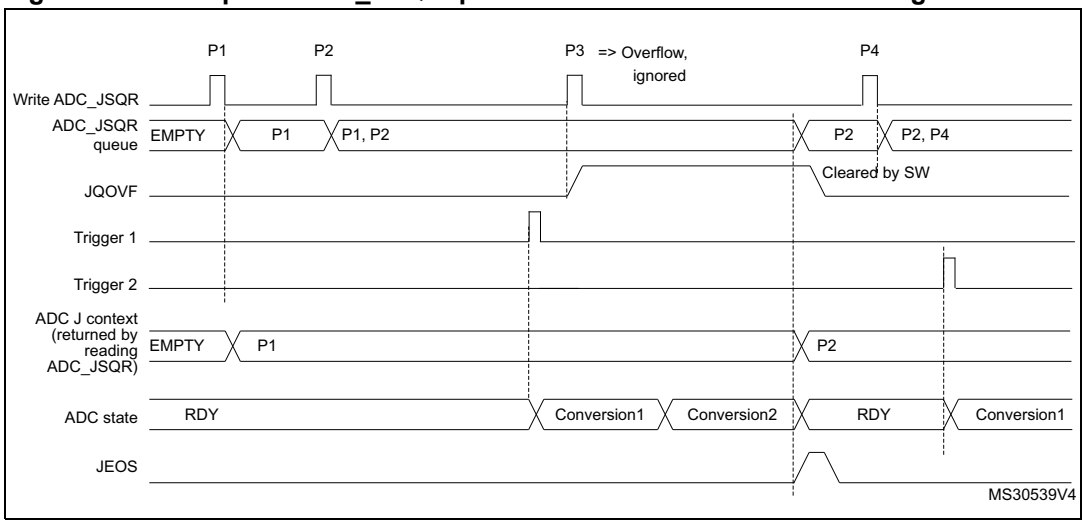
The *Figure 301* and *Figure 302* show the behavior of the context queue if an overflow occurs before or during a conversion.

Figure 301. Example of ADC_JSQR queue of context with overflow before conversion



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

Figure 302. Example of ADC_JSQR queue of context with overflow during conversion



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

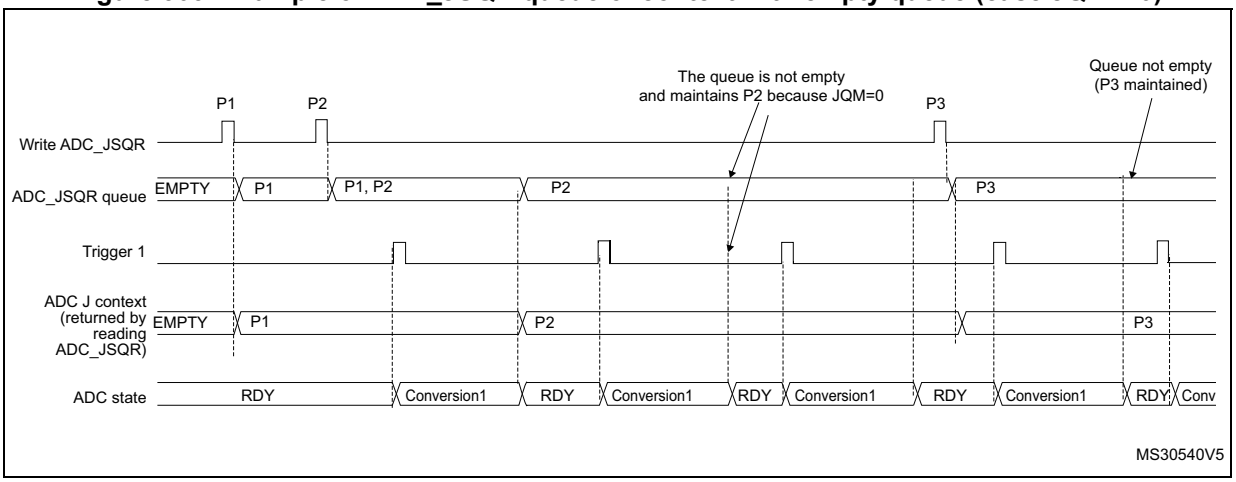
It is recommended to manage the queue overflows as described below:

- After each P context write to the ADC_JSQR register, the JQOVF flag shows if the write has been ignored or not (an interrupt can be generated).
- Avoid queue overflows by writing the third context (P3) only once the flag JEOS of P1 is generated, then the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

Queue of context: Behavior when the queue becomes empty

Figure 303 and *Figure 304* show the behavior of the context queue when the queue becomes empty in both cases JQM = 0 or 1.

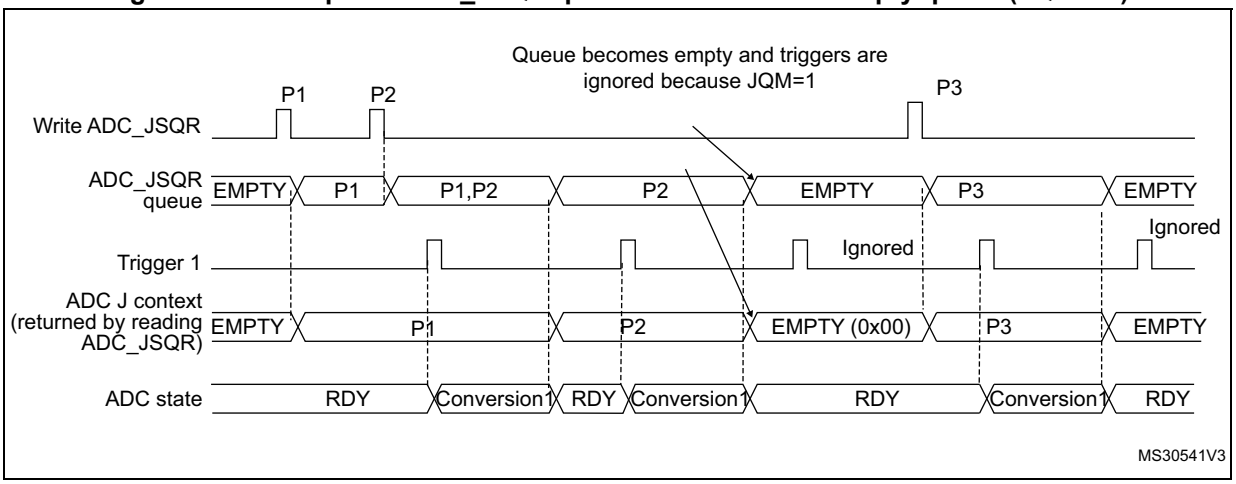
Figure 303. Example of ADC_JSQR queue of context with empty queue (case JQM = 0)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Note: When writing P3, the context changes immediately as the context of P2 is already converted (JEOS is generated). However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately. It is recommended to program the new context before the JEOS timing, otherwise the previous context is used.

Figure 304. Example of ADC_JSQR queue of context with empty queue (JQM = 1)

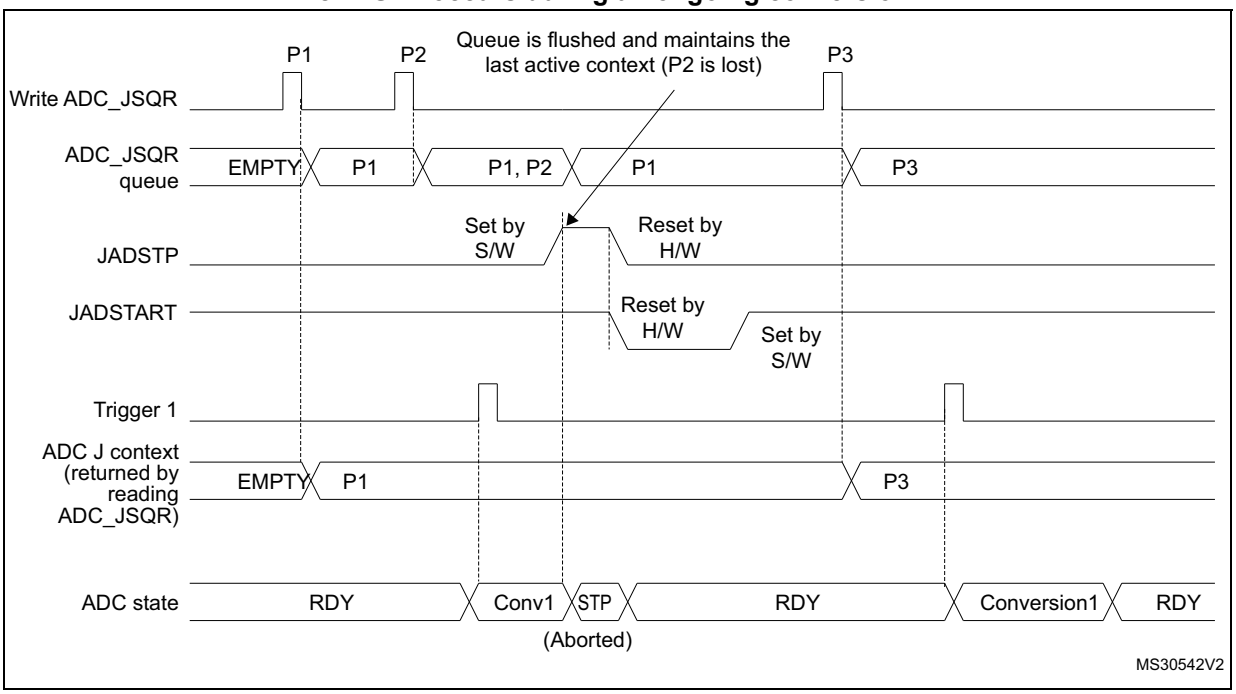


1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Flushing the queue of context

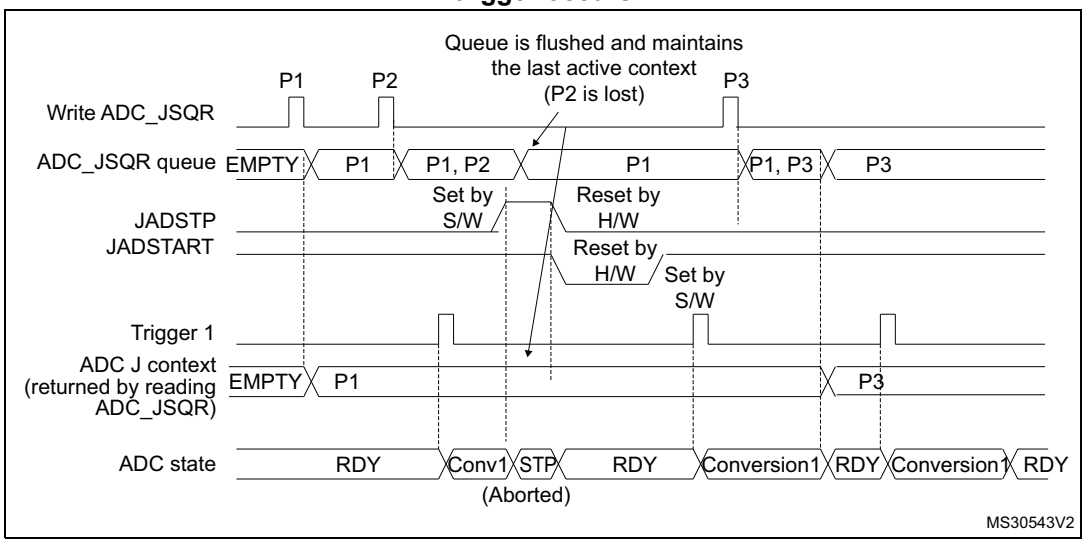
The figures below show the behavior of the context queue in various situations when the queue is flushed.

Figure 305. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion.



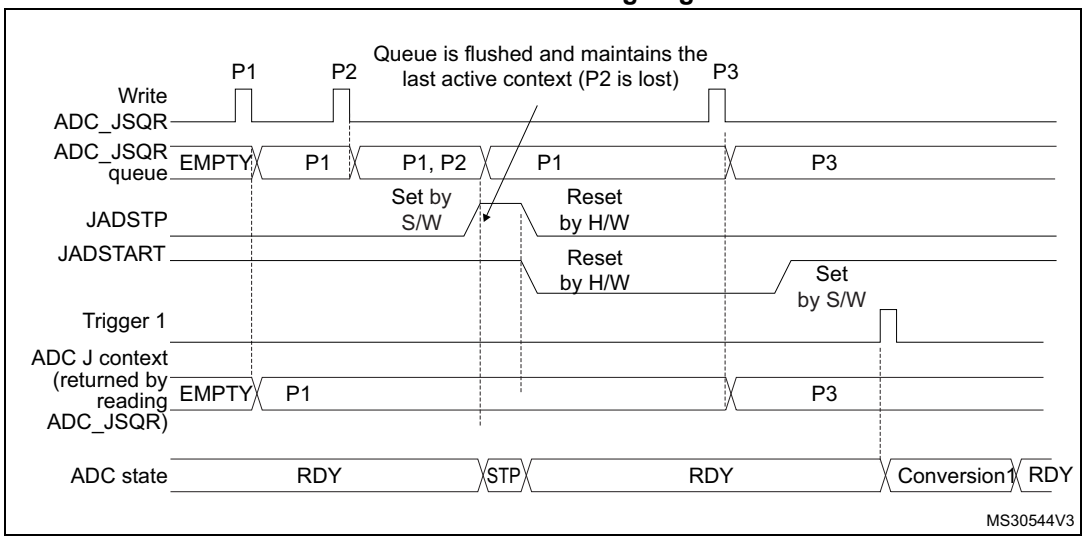
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 306. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion and a new trigger occurs



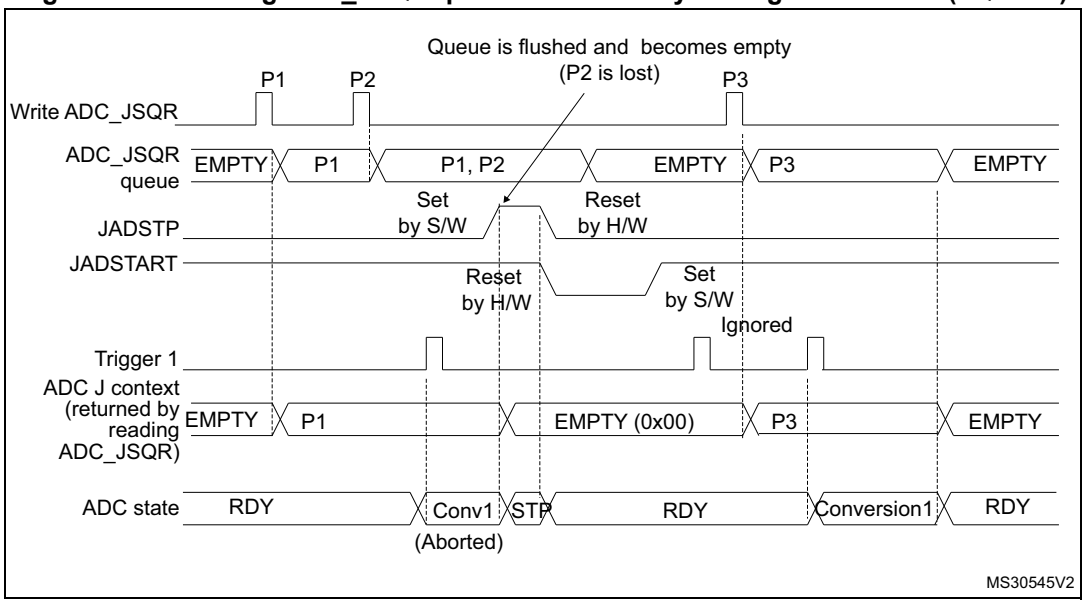
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 307. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs outside an ongoing conversion



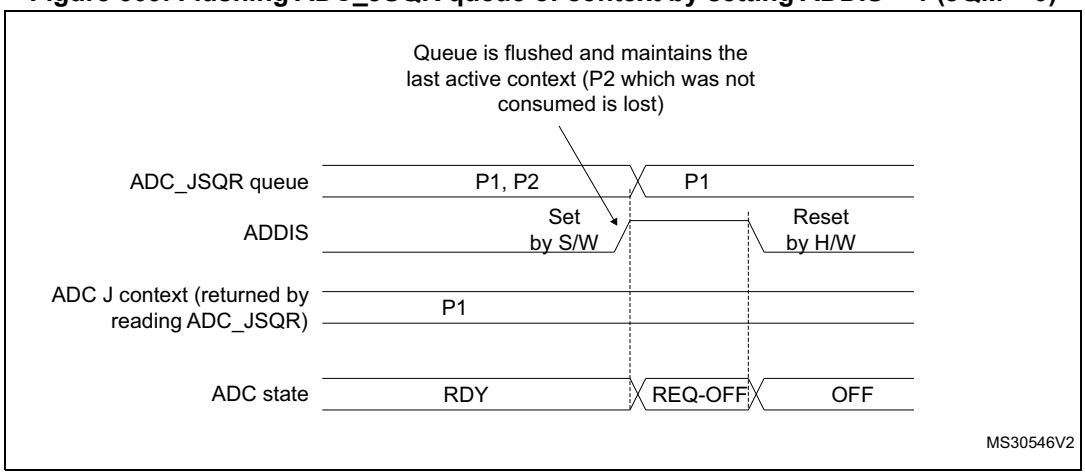
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 308. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 1)



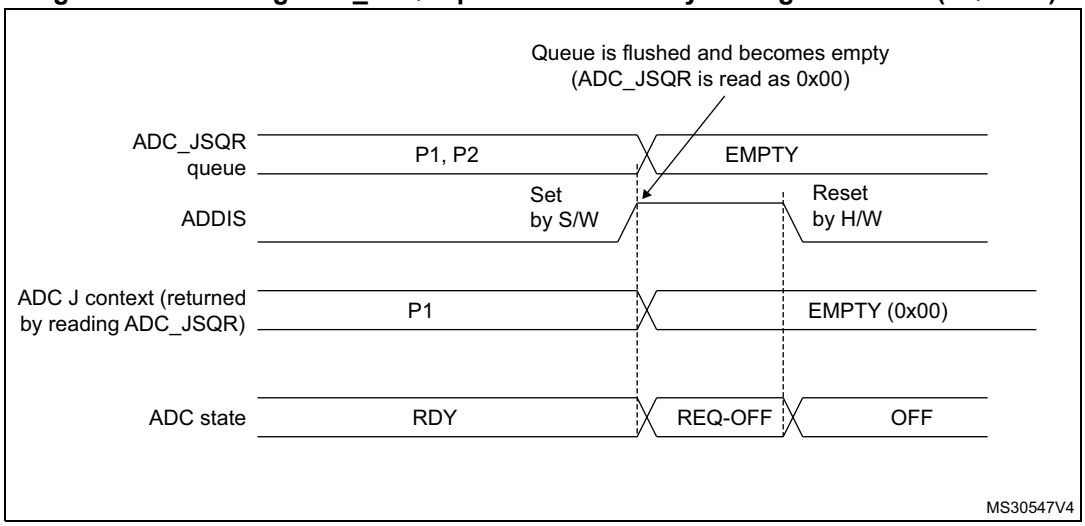
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 309. Flushing ADC_JSQR queue of context by setting ADDIS = 1 (JQM = 0)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 310. Flushing ADC_JSQR queue of context by setting ADDIS = 1 (JQM = 1)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Queue of context: Clear the context

- Setting JQDIS to 0 and then to 1 clears the context.
- Setting JADSTP to 0, writing a new context, and then setting JADSTART to 1 clears the previous context.

Disabling the queue

It is possible to disable the queue by setting the JQDIS bit of the ADC_CFGR1 register.

To clear the context queue, write 0 to JQDIS first, then write 1.

25.4.24 Programmable resolution (RES) - fast conversion mode

Faster conversions can be performed by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the RES[1:0] control bits. *Figure 315*, *Figure 316*, *Figure 317* and *Figure 318* show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to *Table 244*.

Table 244. T_{SAR} timings depending on resolution

RES (bits)	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at $F_{adc_ker_ck}$ = 32 MHz	T_{CONV} (ADC clock cycles) (with Sampling time= 2.5 ADC clock cycles)	T_{CONV} (ns) at $F_{adc_ker_ck}$ = 32 MHz
12	13.5 ADC clock cycles	421.875 ns	16 ADC clock cycles	500.0 ns
10	11.5 ADC clock cycles	359.375 ns	14 ADC clock cycles	433.33 ns

Table 244. T_{SAR} timings depending on resolution (continued)

RES (bits)	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at $F_{adc_ker_ck}$ = 32 MHz	T_{CONV} (ADC clock cycles) (with Sampling time= 2.5 ADC clock cycles)	T_{CONV} (ns) at $F_{adc_ker_ck}$ = 32 MHz
8	9.5 ADC clock cycles	296.875 ns	12 ADC clock cycles	366.67 ns
6	7.5 ADC clock cycles	156.25 ns	10 ADC clock cycles	300 ns

25.4.25 End of conversion and end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application at the end of each regular conversion (EOC) event and injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC_DR register. An interrupt can be generated if the EOCIE bit is set. The software can clear the EOC flag either by programming it to 1 or by reading ADC_DR. If ADC_DR FIFO is not emptied, the EOC flag is set again and an interrupt can be generated again.

The ADC sets the JEOC flag as soon as new injected conversion data is available in one of the ADC_JDRy registers. An interrupt can be generated if the JEOCIE bit is set. The software can clear the JEOC flag either by programming it to 1 or by reading the corresponding ADC_JDRy register.

The ADC also notifies the end of the sampling phase by setting the EOSMP flag (for regular conversions only). The EOSMP flag is cleared by software by programming it to 1. An interrupt can be generated if the EOSMPIE bit is set.

25.4.26 End of conversion sequence (EOS, JEOS)

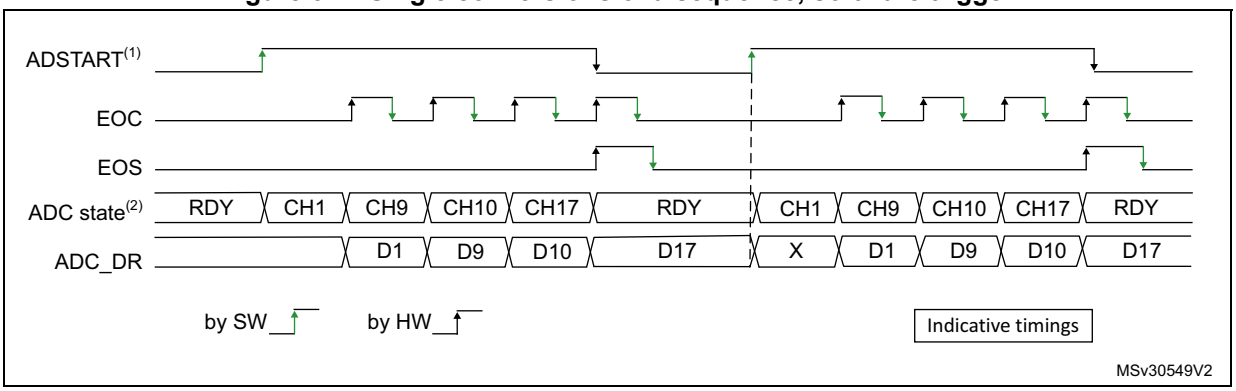
The ADC notifies the application at the end of each regular sequence (EOS) and injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the conversion of the last data of the regular sequence is complete. An interrupt can be generated if the EOSIE bit is set. The EOS flag is cleared by the software by programming it to 1.

The ADC sets the JEOS flag as soon as the conversion of the last data of the injected sequence is complete. An interrupt can be generated if the JEOSIE bit is set. The JEOS flag is cleared by the software by programming it to 1.

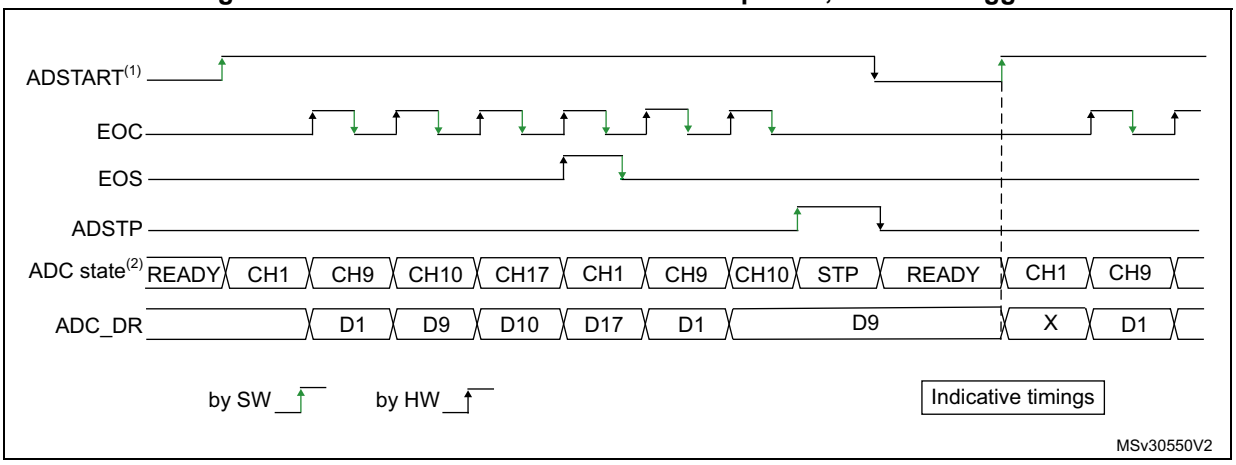
25.4.27 Timing diagram examples (single/continuous modes, hardware/software triggers)

Figure 311. Single conversions of a sequence, software trigger



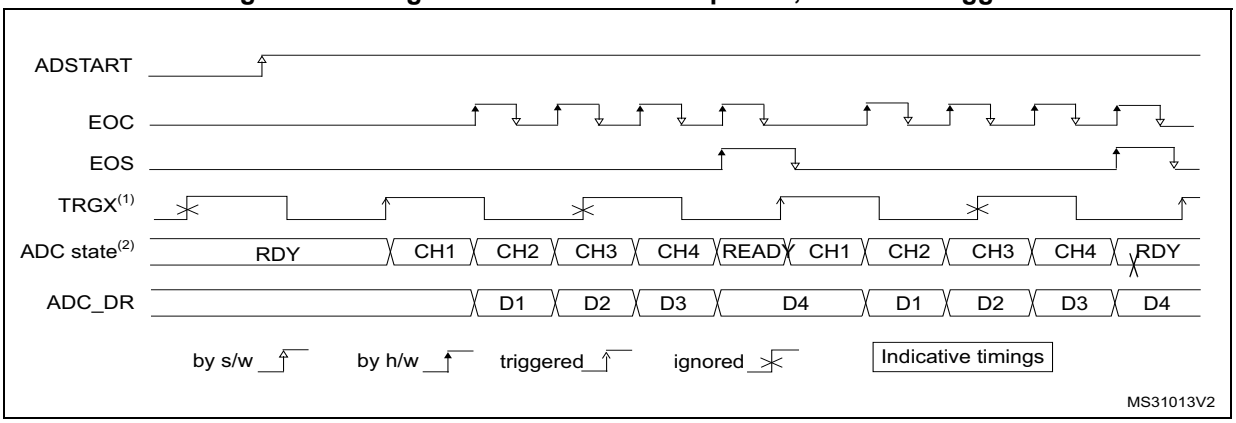
1. EXTEN[1:0] = 00, CONT = 0
2. Channels selected = 1, 9, 10, 17; AUTDLY = 0.

Figure 312. Continuous conversion of a sequence, software trigger



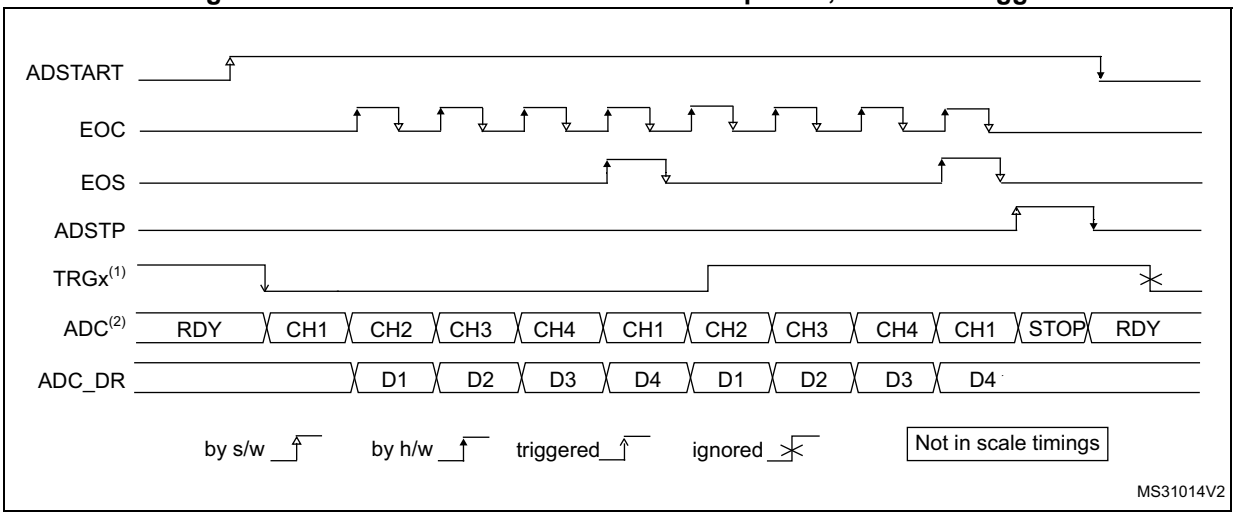
1. EXTEN[1:0] = 00, CONT = 1
2. Channels selected = 1, 9, 10, 17; AUTDLY = 0.

Figure 313. Single conversions of a sequence, hardware trigger



1. TRGx (overfrequency) is selected as trigger source, EXTEN[1:0] = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

Figure 314. Continuous conversions of a sequence, hardware trigger



1. TRGx is selected as trigger source, EXTEN[1:0] = 10, CONT = 1.
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

25.4.28 Data management

Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS[a], JLSHIFT[a], USAT, SSAT, POSOFF)

Data and alignment

At the end of each regular channel conversion (when an EOC event occurs), the result of the converted data is stored in the 32-bit wide ADC_DR data register with FIFO (when OVRMOD = 0). Since the data register has eight levels of FIFO, the EOC flag is raised again as far as the FIFO is not emptied.

At the end of each injected channel conversion (when a JEOC event occurs), the result of the converted data is stored in the 32-bit wide corresponding ADC_JDRy data register.

The OVSS[3:0], LSHIFT[3:0], JOVSS[3:0], and JLSHIFT[3:0] bitfields in the ADC_CFGR2 register select the alignment of the data stored after the conversion. By default, data are right-aligned. Refer to *Figure 315*, *Figure 316*, *Figure 317*, and *Figure 318* for examples of data alignment.

Note: The data can be aligned in normal and oversampling mode. The OVSS[3:0] bitfield is only effective in oversampling mode.

JOVSS[3:0] and JLSHIFT[3:0] is reserved for injected conversions. Their usage is the same as OVSS[3:0] and LSHIFT[3:0], unless otherwise specifically described.

Offset

An unsigned offset value y (y = 1, 2, 3, 4) can be applied to a channel by programming a value different from 0 in the OFFSET[21:0] bitfield of the ADC_OFRy register. The channel to which the offset is applied is programmed to the OFFSETy_CH[4:0] bits of ADC_OFCFGRy register. The offset can be positive or negative depending on the value of POSOFF bit. When POSOFF is cleared, the converted value is subtracted by the user-defined offset written in OFFSET[21:0] bits. The result can be a negative value. The read data is consequently signed and the SEXT bit represents the extended sign value.

The offset value must be lower than the maximum conversion value (for example the maximum offset value is 0xFFF in 12-bit mode).

The offset can be used to convert unsigned data to signed data (for example the offset value is equal to 0x800 in 12-bit mode).

The offset correction is also supported in oversampling mode. In this mode, the offset is subtracted when all the oversampling operations are complete.

Note: If the OFFSET channel is selected but the offset value is 0x0000, this offset programming is ignored. If the same OFFSET channel is programmed in different registers multiple times, the first programmed register takes priority.

Table 250 describes how the computation is performed for all the possible ADC resolutions and offset settings.

a. Only for products that include this bit.

Table 245. Offset computation versus data resolution

Resolution (RES[1:0] bits)	Subtraction between raw converted data and offset		Result	Comments
	Raw converted Data, left aligned	Offset		
00: 12-bit	DATA[11:0]	OFFSET[11:0]	Signed or unsigned 24-bit data, right aligned to [11:0]	-
01: 10-bit	DATA[11:2],00	OFFSET[11:2]	Signed or unsigned 24-bit data, right aligned to [9:0]	The user must configure OFFSET[1:0] to 0b00.
10: 8-bit	DATA[11:4],0000	OFFSET[11:4]	Signed or unsigned 24 bit data right aligned to [7:0]	The user must configure OFFSET[3:0] to 0b0000.
11: 6-bit	DATA[11:6],000000	OFFSET[11:6]	Signed or unsigned 24 bit data right aligned to [5:0]	The user must configure OFFSET[5:0] to 0b000000.

Figure 315, *Figure 316*, *Figure 317* and *Figure 318* show alignments for signed and unsigned data. JOVSS and JLSHIFT are not described here, but they behave the same as OVSS and LSHIFT.

Figure 315. Right alignment (offset disabled, unsigned value)

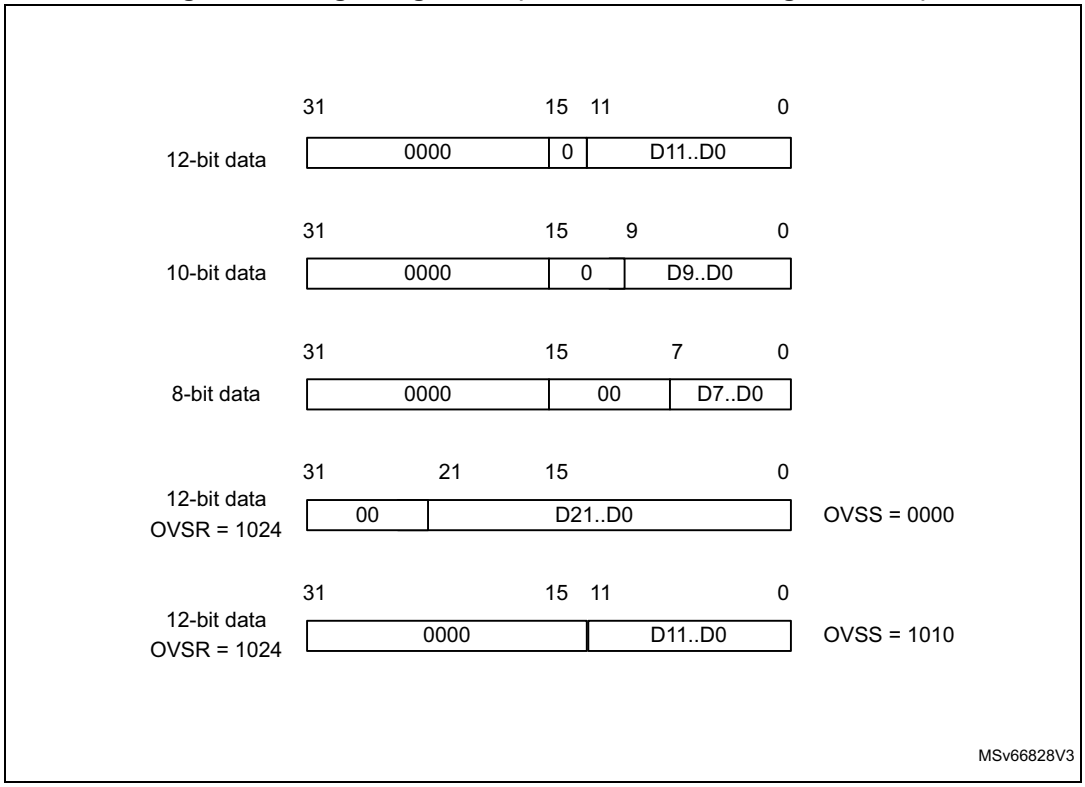


Figure 316. Right alignment (offset enabled, signed value)

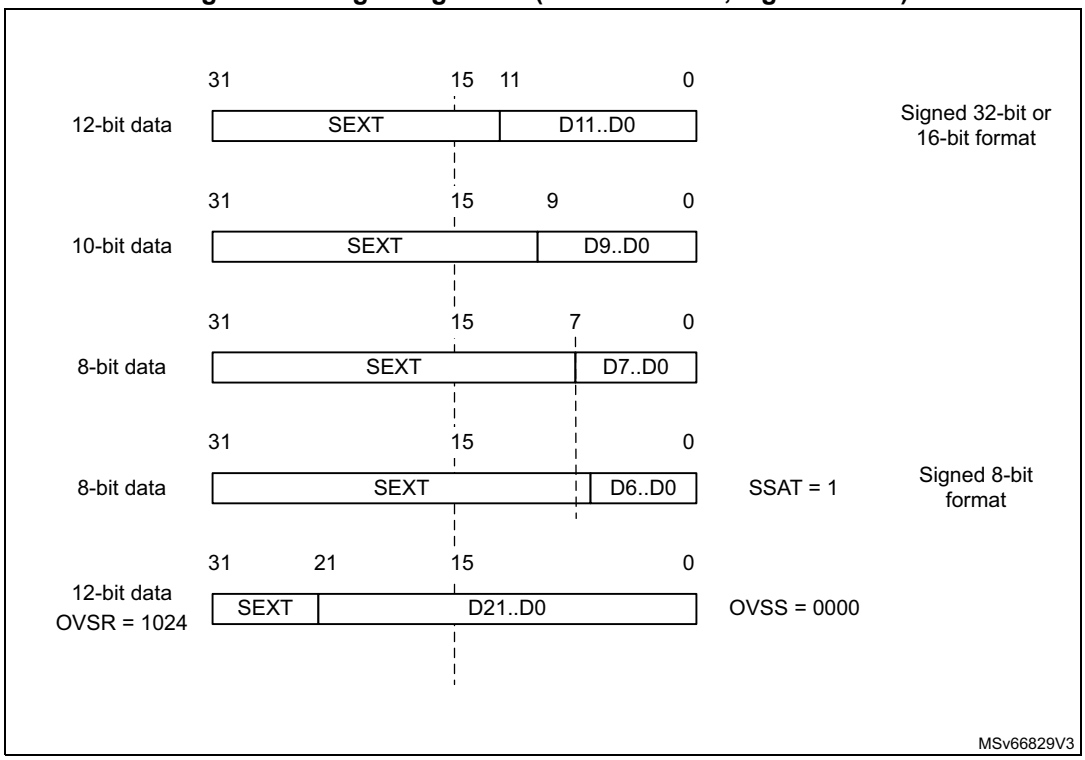


Figure 317. Left alignment (offset disabled, unsigned value)

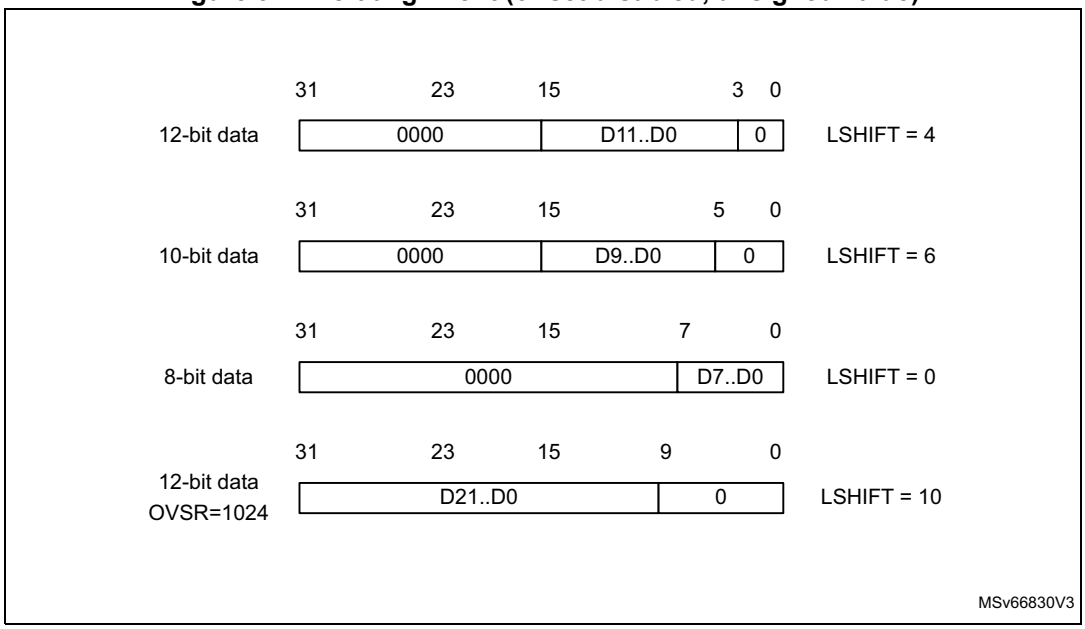
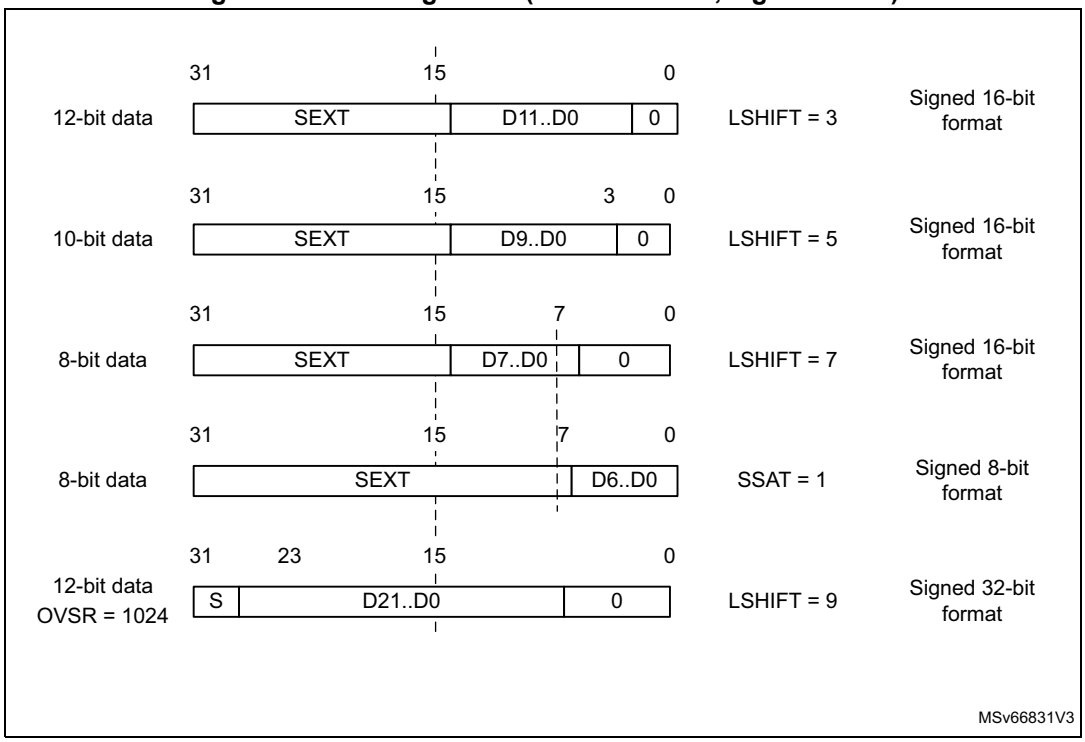


Figure 318. Left alignment (offset enabled, signed value)



Management of signed and unsigned saturation format (SSAT, USAT)

The offset correction might result in the data width to be wider than the original data.

To limit the original data width, the data saturation can be enabled through the SSAT and USAT bits of the ADC_OFCFGRy register.

Unsigned 12-bit data can be extended to 13-bit signed data by using an offset value different from 0x800.

The original data width can be preserved by setting SSAT bit to limit the data width to 12 bits.

Unsigned data can be saturated to the original data width by setting the USAT bit.

Table 246 shows the sign-extended data format corresponding to different resolutions.

Table 246.12-bit data formats

SSAT	USAT	Format	Data range (offset =0x800)
0	0	Sign-extended 13-bit significant data: SEXT[31:13] DATA[12:0]	0x0000 07FF - 0xFFFF F800
1	0	Sign-extended 12-bit significant data: SEXT[31:12] DATA[11:0]	0x7FF - 0x800

Table 246.12-bit data formats

SSAT	USAT	Format	Data range (offset =0x800)
0	1	Unsigned saturation 12-bit signification data: DATA[11:0]	0xFFF - 0x000
1	1	Reserved	-

Table 247 provides numerical examples for four different offset values.

Table 247. Numerical examples for 32-bit or 16-bit format (POSSPFF = 0)

Raw conversion result	Offset value	Result SSAT = 0 USAT = 0	Result SSAT = 0 USAT = 1	Result SSAT = 1 USAT = 0
0xFFF	0x800	0x0000 07FF	0x07FF	0x07FF
0x800		0x0000 0000	0x0000	0x0000
0x000		0xFFFF F800	0x0000	0xF800
0xFFF	0x820	0x0000 07DF	0x07DF	0x07DF
0x800		0xFFFF FFE0	0x0000	0xFFE0
0x000		0xFFFF F7E0	0x0000	0xF7E0
0xFFF	0x7E0	0x0000 081F	0x081F	0x07FF
0x800		0x0000 0020	0x0020	0x0020
0x000		0xFFFF F820	0x0000	0xF820
0xFFF	0x020	0x0000 0FDF	0x0FDF	0x0FDF
0x800		0x0000 07E0	0x07E0	0x07E0
0x000		0xFFFF FFE0	0x0000	0xFFE0

Caution: SSAT must not be used in conjunction with USAT. No hardware check is performed to ensure that this condition is respected.

Gain compensation

When the GCOMP bit is set in the ADC_GCOMP register, the gain compensation is activated on all the converted data. After each conversion, data is calculated using the following formula.

$$DATA = DATA(\text{adc result}) \times (GCOMPCOEFF) / 4096$$

As GCOMPCOEFF can be programmed from 0 to 16383, the actual gain compensation factor can range from 0 to 3.999756.

Before storing the resulting data in RDATAx or JDATAx bitfields, the LSB - 1 value is evaluated to round up the data and minimize the error.

The gain compensation is also effective for the oversampling. When the gain compensation is used in oversampling mode, the gain calculation is performed after the accumulation and

right-shift operations to minimize the power consumption (the gain calculation is done only once and not at each conversion).

The internal multiplier width is 32 bits and the input data width for the gain compensation must be less than 18 bits. When using oversampling with injected and regular conversion mode, the ROVSM bit of the ADC_CFGR2 register must be set to resume the pending conversion with the correct value.

When the gain compensation is activated, the ADC data latency is increased by one clock cycle. In continuous and scan conversion modes, the ADC conversion rate does not change.

ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) issues a notification that the regular converted data has not been read (by the CPU or the DMA) before the ADC_DR FIFO (eight stages) is overflowed, or adc_hclk clock is too slow to manage the data. If the CPU clears the EOC flag without reading the data register, the OVR flag may still be triggered.

The OVR flag is set when a new conversion completes while ADC_DR as single register or FIFO output is full. An interrupt is generated if the OVRIE bit is set.

When an overrun condition occurs, the ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting ADSTP. The OVR flag is cleared by software by programming it to 1.

Data can be configured to be preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD = 0

  The overrun event preserves the data register from being overwritten: the old data are maintained up to ADC_DR FIFO depth (eight data) and the new conversion is discarded and lost. If OVR remains at 1, further conversions occur but the result data is also discarded. The FIFO can be emptied through ADC_DR read access by the CPU while the OVR flag is set.
- OVRMOD = 1

  The data register is overwritten with the last conversion result and the previous unread data is lost. In this mode, ADC_DR FIFO is disabled. If OVR remains at 1, any further conversion is performed normally and the ADC_DR register always contains the latest converted data. In this mode, DMA burst mode nor ADF output mode cannot be supported.

Figure 319. Example of overrun (OVRMOD = 0)

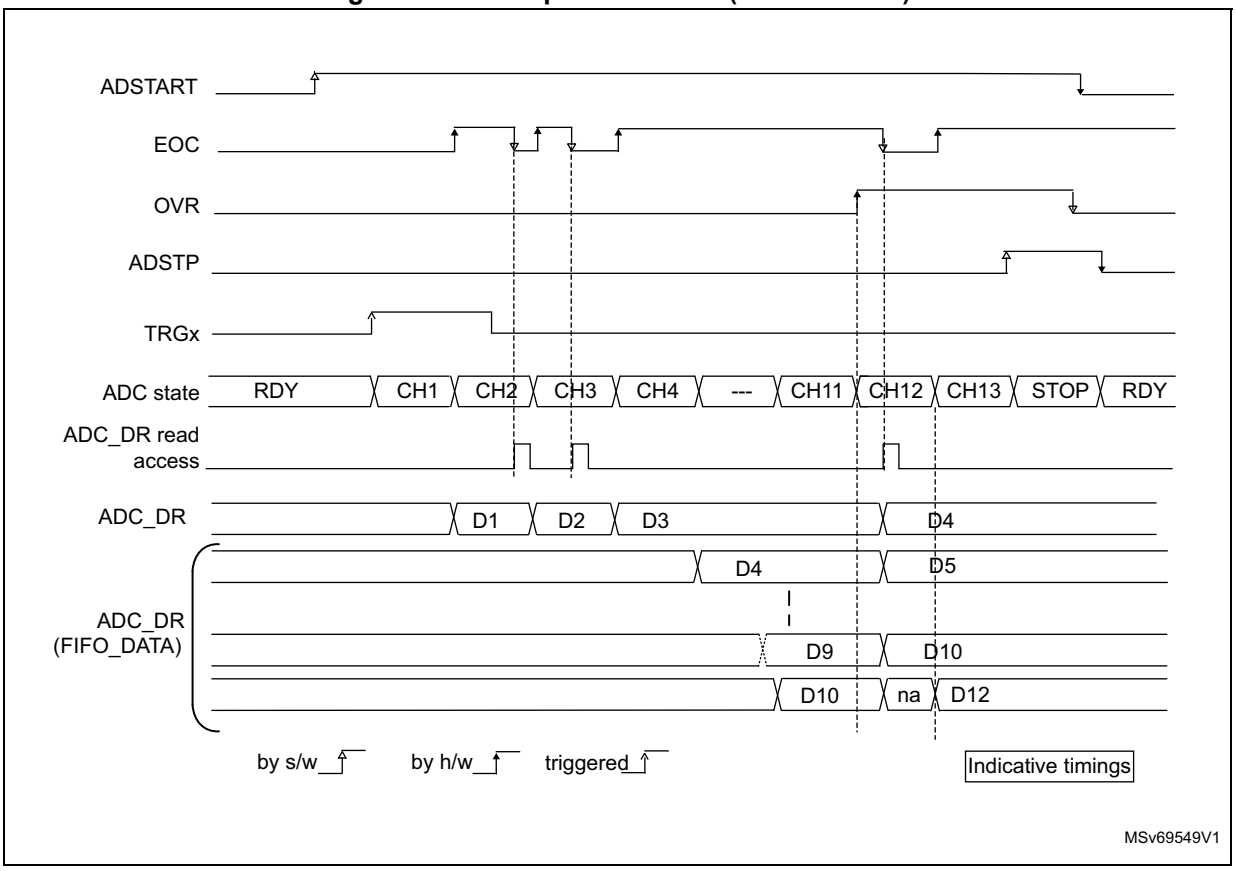
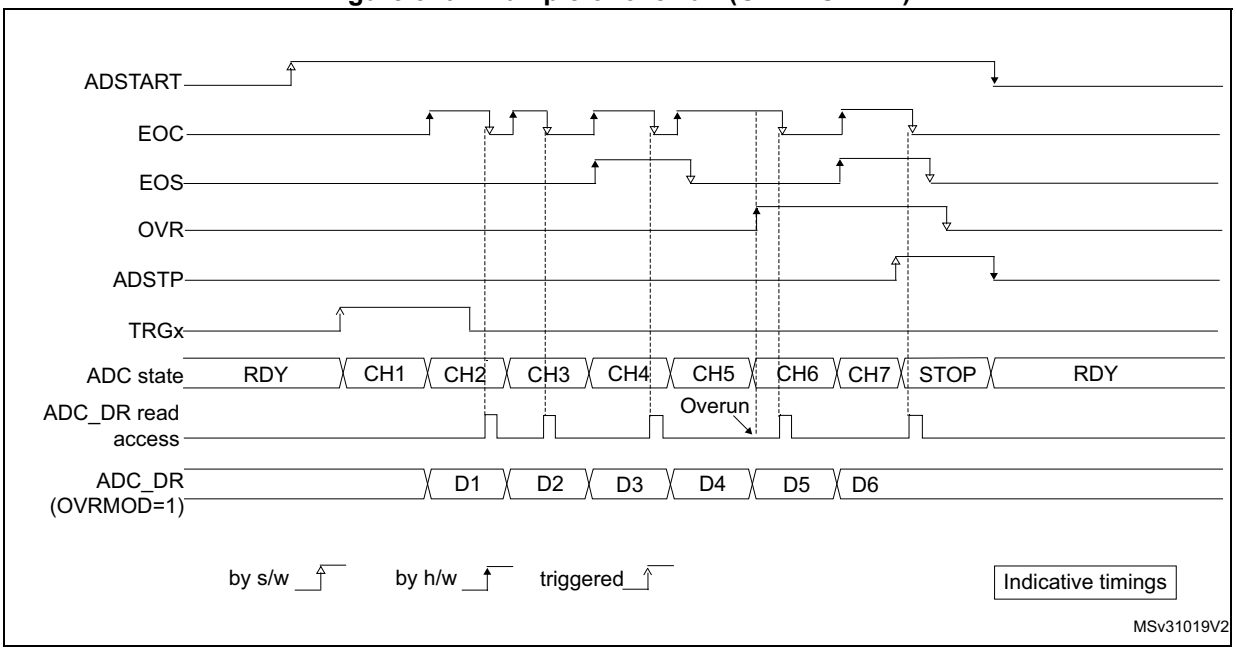


Figure 320. Example of overrun (OVRMOD = 1)



Note: Even if overrun may occur for injected channels, there is no overrun detection on these channels since there is a dedicated data register for each of the four injected channels.

Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD must be configured to 0 to manage overrun events or FIFO overflow as errors.

Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and the OVR flag must be ignored by the software. An overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion.

Managing conversions using the DMA

When the DMA mode is enabled (DMNGT[1:0] = 01 or 11 in the ADC_CFGR1 register in single ADC mode), a DMA request is generated after each channel conversion. This allows the transfer of the converted data from the ADC_DR register to the destination location configured in the DMA.

Despite this, if an overrun occurs (OVR = 1) because the DMA cannot serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA, which means that all the data transferred to the RAM can be considered as valid. (OVRMOD = 0).

When OVRMOD = 1, even if DMA does not transfer the data immediately, new data overwrite ADC_DR, causing DMA to potentially skip data. As a result, some conversion results may be missed in the data transferred to RAM.

Depending on the configuration of the OVRMOD bit, the data are either preserved or overwritten (refer to *Section : ADC overrun (OVR, OVRMOD)*). If OVRMOD = 0 (data preserved), DMA transfer requests are blocked.

DMA can be configured through the DMNGT[1:0] bitfield of the ADC_CFGR1 register. Refer to *Section 25.4.34: Dual ADC modes* for information on DMA usage with dual ADC mode.

- DMA one-shot mode (DMNGT[1:0] = 01)

  This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMNGT[1:0] = 11)

  This mode is suitable when programming the DMA when the number of transfers is unknown.

DMA one-shot mode (DMNGT[1:0] = 01)

In one-shot mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (that is when DMA_LACK interrupt occurs, see *Direct-memory controller* section) even if a new conversion has been started.

When DMA transfers are complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- All ongoing conversions are aborted with the partial result discarded.
- The scan sequence is stopped and reset.
- The DMA and ADC are stopped.

DMA circular mode (DMNGT[1:0] = 11)

In circular mode, the ADC generates a DMA transfer request each time a new conversion data is available in the data register. This allows the configuration of the DMA in circular mode to handle a continuous analog input data stream.

DMA single-transfer or burst mode with FIFO (OVRMOD = 0)

The output data register features an eight-stage FIFO. Two different DMA requests are generated in parallel. When a data is available, an “SREQ single request” is generated. When four data are available, a “BREQ burst request” is generated. The DMA can be programmed either in single-transfer mode or in burst mode (four beats). The appropriate request line is selected by the DMA according to the mode selected. Refer to *Section Direct memory access controller* for further information.

Due to the existence of FIFO, DMA requests are generated until the FIFO is emptied. To stop DMA requests even when the FIFO is not emptied, set the ADSTP bit.

DMA for dual mode

Refer to *Section 25.4.34: Dual ADC modes*.

25.4.29 Managing conversions using the ADF

The ADC conversion results can be transferred directly to the ADF.

In this case, the DMNGT[1:0] bits must be set to 10.

The ADC transfers the 16 least significant bits of the regular data register to the ADF through adcx_dat[15:0] bus, which in turn resets the EOC flag once the transfer is effective.

The data format must be in 16-bit signed format:

ADC_DR[31:16] = Don’t care

ADC_DR[15] = Sign

ADC_DR[14:0] = Data

Any value above the 16-bit signed format is truncated.

To obtain a 16-bit signed format, the OFFSET function needs to be activated.

In this mode, OVRMOD = 1 is not supported.

25.4.30 Dynamic low-power features

Autodelayed conversion mode (AUTDLY)

The ADC implements an autodelayed conversion mode controlled by the AUTDLY configuration bit in ADC_CFGR1. Autodelayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there is a risk of encountering an ADC overrun.

Regular conversion mode

When AUTDLY is set, a new conversion can start if the following conditions are met:

- The ADC_DR register is read or the EOC bit is cleared (see *Figure 321*).

This is a way to automatically adapt the speed of the ADC to the speed of the system that reads the data and prevent overrun errors.

The delay is inserted after each regular conversion.

Injected conversion mode

When AUTDLY is set, a new conversion can start only if the JEOS bit has been cleared (see *Figure 322*). Since there are four ADC_JDRy registers, the autodelay is performed at the end on the injected conversion sequence.

There is no delay inserted between each conversion of the injected sequence, except after the last one.

A hardware trigger event occurring during this delay is ignored.

No delay is inserted between regular and injected conversions. The autodelay for regular and injected modes is managed separately.

Caution: The behavior is slightly different in autoinjection mode (JAUTO = 1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see *Figure 325*).

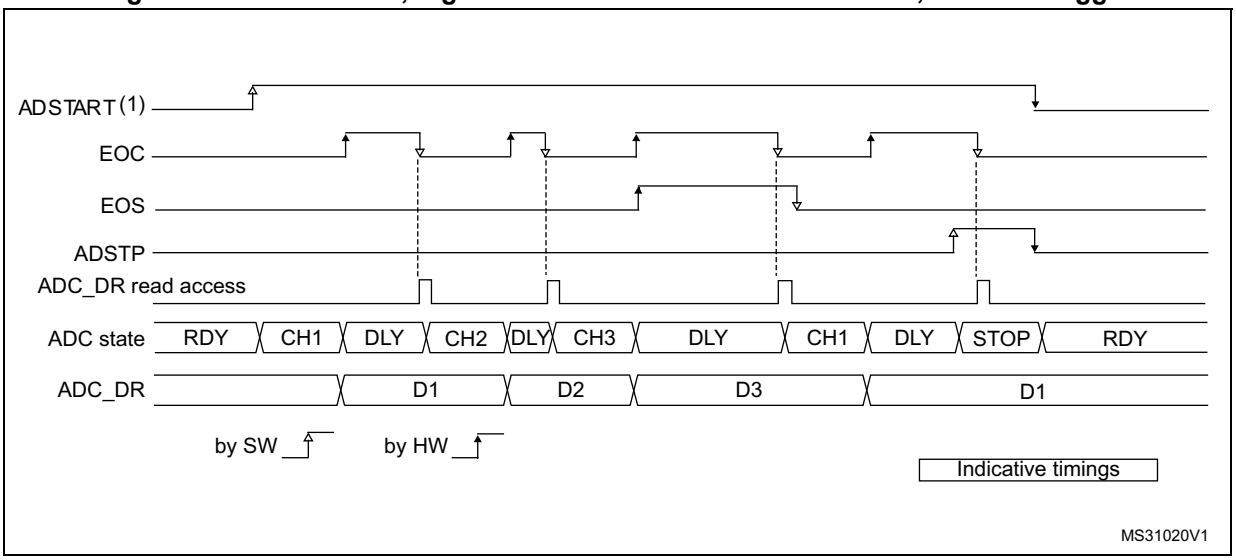
In autoinjection mode, there is no delay between the last regular conversion and the first injected conversion. The user must clear the last EOC flag before clearing the JEOS flag, otherwise the next sequence might start before EOC flag is cleared.

In autodelay mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

In autodelay mode with JAUTO = 0, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

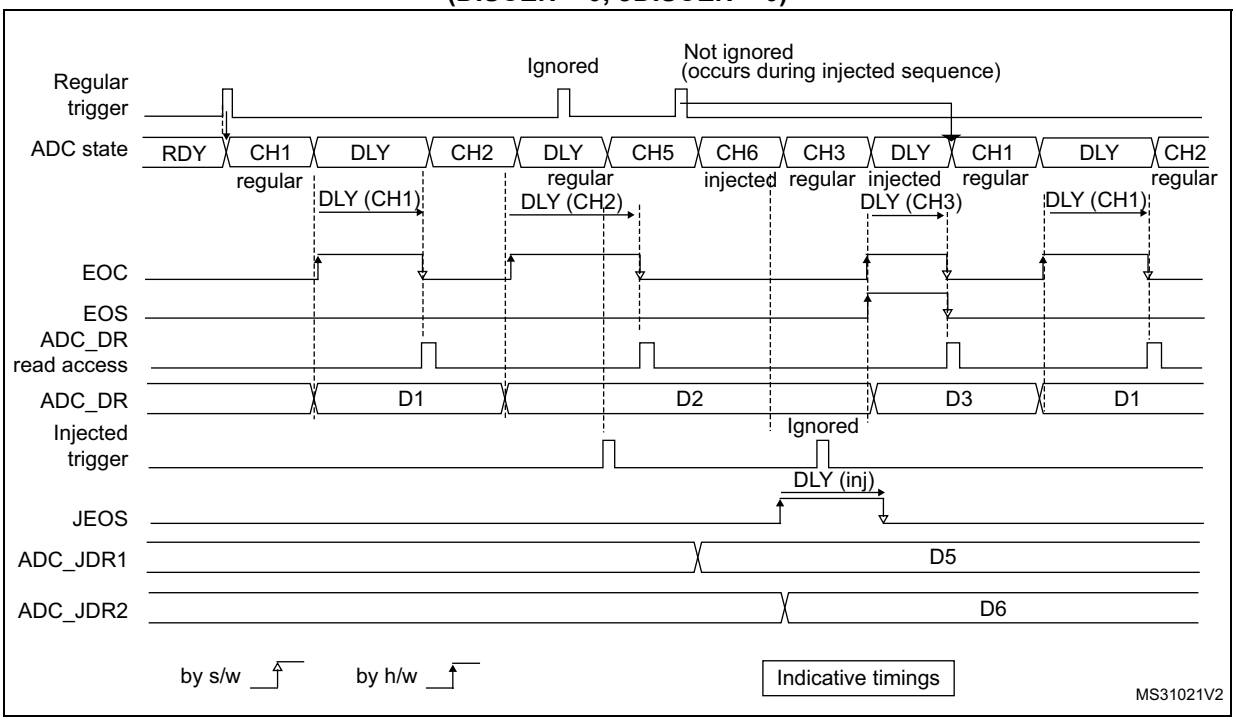
Note: AUTDLY mode is not supported with injected queue, SMPTRIG, or BULB mode.

Figure 321. AUTDLY = 1, regular conversion in continuous mode, software trigger



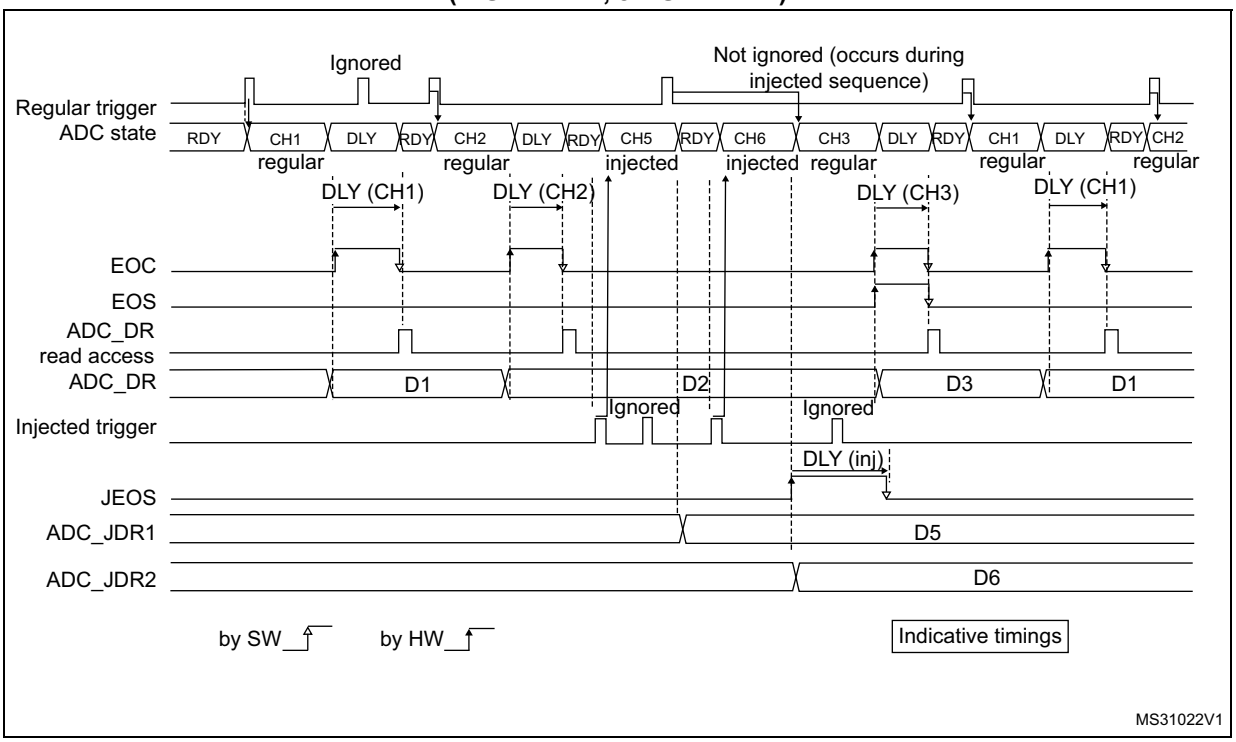
1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, CHANNELS = 1, 2, 3.
3. Injected configuration DISABLED.

Figure 322. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0)



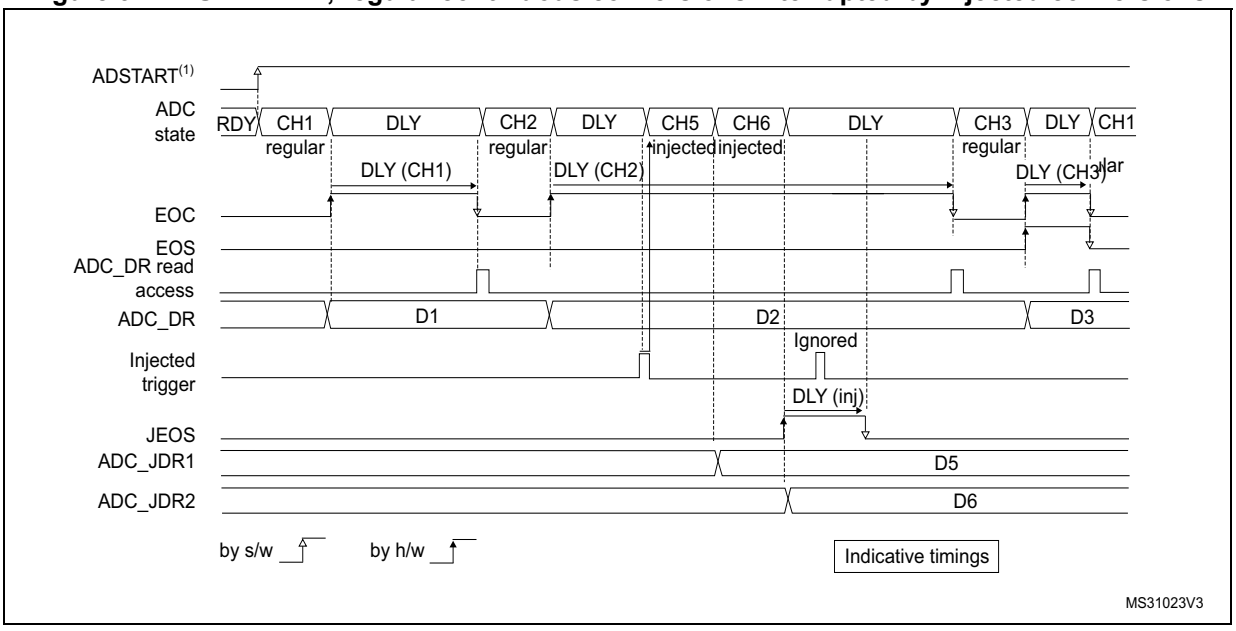
1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 01 (hardware trigger), CONT = 0, DISCEN = 0, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 0, CHANNELS = 5, 6.

Figure 323. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1)



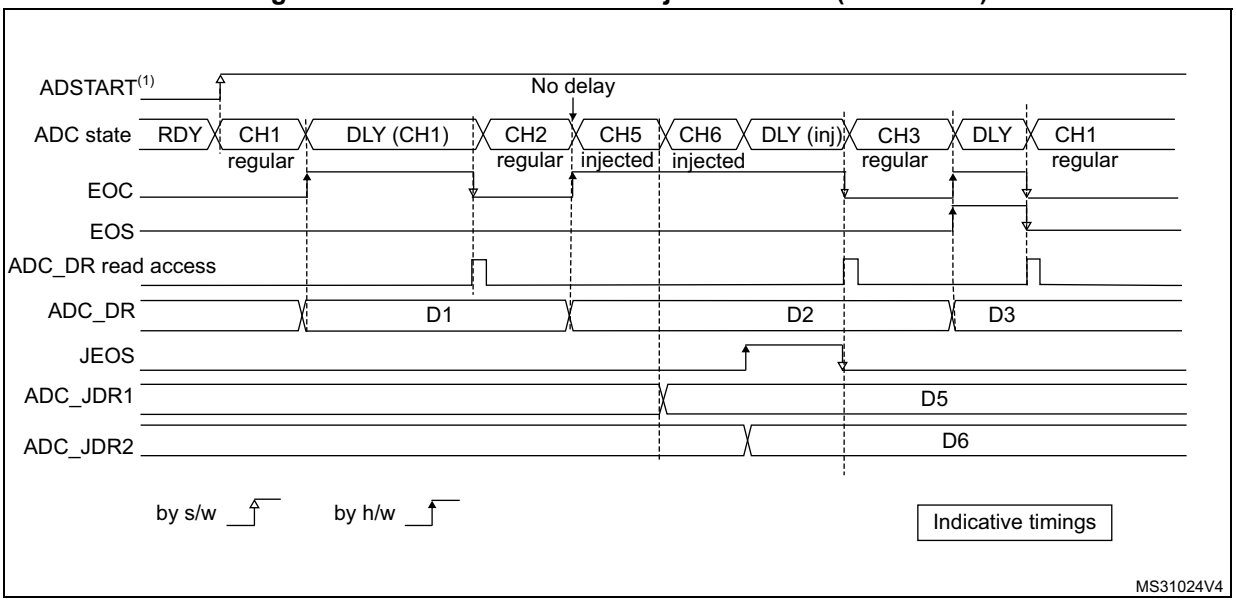
1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 01 (hardware trigger), CONT = 0, DISCEN = 1, DISCNUM = 1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 1, CHANNELS = 5, 6.

Figure 324. AUTDLY = 1, regular continuous conversions interrupted by injected conversions



1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 0, CHANNELS = 5, 6.

Figure 325. AUTDLY = 1 in autoinjection mode (JAUTO = 1)



1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2.
3. Injected configuration: JAUTO = 1, CHANNELS = 5, 6.

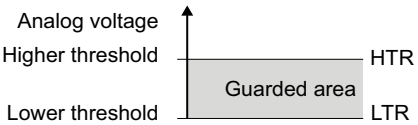
25.4.31 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)

Table 248. Analog window watchdog features

Feature	AWD1	AWD2	AWD3
Enable control	AWD1EN or JAWD1EN	ADC_AWD2CR	ADC_AWD3CR
Channel selection	AWD1CH[4:0]/AWD1SGL	ADC_AWD2CR	ADC_AWD3CR
Channel selection limitation	One or all channels	Any number of channels	
	Regular or injected	-	-
Threshold control	ADC_AWD1LTR/ ADC_AWD1HTR	ADC_AWD2LTR/ ADC_AWD2HTR	ADC_AWD3LTR/ ADC_AWD3HTR
Filter configuration	AWDFILT[2:0] in ADC_AWDyHTR		
Interrupt	AWDy in ADC_ISR		

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Figure 326. Analog watchdog guarded area



AWDy flag and interrupt

An interrupt can be enabled for each of the three analog watchdogs by setting AWDyIE in the ADC_IER register (y = 1, 2, 3).

AWDy (y = 1, 2, 3) flag is cleared by software by writing 1 to it, or by setting ADDIS.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

AWD source channel register

The software can identify which ADC channel has generated the AWDy flag by reading the AWDS_CH[19:0] bitfield of the ADC_AWDySR register: AWDS_CH*i* bit is set to indicate that channel *i* has generated the flag. It is cleared by writing 1 to it or by disabling the ADC through setting the ADDIS bit.

Description of analog watchdog 1

The AWD analog watchdog 1 is enabled by setting the AWD1EN or JAWD1EN bit in the ADC_CFGR1 register. This watchdog monitors whether either one selected channel or all enabled channels remain within a configured voltage range (window).

Table 249 shows how the ADC_CFGR1 registers must be configured to enable the analog watchdog on one or more channels.

Table 249. Analog watchdog 1 channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit	JAWD1EN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single(1) injected channel	1	0	1
Single(1) regular channel	1	1	0
Single(1) regular or injected channel	1	1	1

1. Selected by the AWD1CH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

The AWD1 analog watchdog flag is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold.

These thresholds are programmed through HTR[22:0] bits of the ADC_AWD1HTR register and LTR[22:0] bits of the ADC_AWD1LTR register for the analog watchdog 1.

The threshold can be up to 23 bits (12-bit resolution with oversampling, OVSR = 1024, offset and gain compensation)

When converting data with a resolution of less than 12 bits (defined by RES[1:0] bits), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data.

Table 250 describes how the comparison is performed for all the possible resolutions and for analog watchdog 1.

Table 250. Analog watchdog 1, 2, 3 comparison

Resolution (RES[1:0] bits)	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned(1)	Thresholds	
00: 12-bit	DATA[11:0]	LTR[22:0] and HTR[22:0]	-
01: 10-bit	DATA[11:2],00	LTR[22:0] and HTR[22:0]	The user must configure LTR[1:0] to 0b00 and HTR[1:0] to 0b00.
10: 8-bit	DATA[11:4],0000	LTR[22:0] and HTR[22:0]	The user must configure LTR[3:0] to 0b0000 and HTR[3:0] to 0b0000.
11: 6-bit	DATA[11:6],000000	LTR[22:0] and HTR[22:0]	The user must configure LTR[5:0] to 0b000000 and HTR[5:0] to 0b000000.

1. Refer to *Section : Gain compensation* for additional details on analog watchdog comparison.

Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDCH[19:0] (of ADC_AWD2CR and ADC_AWD3CR).

The corresponding watchdog is enabled when any bit of AWDCH[19:0] (of ADC_AWD2CR and ADC_AWD3CR) is set.

The threshold can be up to 23 bits (12-bit resolution with oversampling, OVSR = 1024 and offset compensation in signed format with gain compensation) and is programmed through the ADC_AWD2HTR, ADC_AWD2LTR, ADC_AWD3HTR and ADC_AWD3LTR registers.

When converting data with a resolution of less than 12 bits (defined by RES[1:0]) bits), the LSBs of the programmed threshold must be kept cleared, the internal comparison being performed on the full 12-bits converted data (left aligned).

ADCx_AWDy_OUT signal output generation

Each analog watchdog is associated with an internal hardware signal ADC_AWDy_OUT (*y* being the watchdog number), which is directly connected to the ETR input (external trigger) of some of the on-chip timers. Refer to the on-chip timers sections to understand how to select the ADC_AWDy_OUT signal as ETR.

ADC_AWDy_OUT is activated when the associated analog watchdog is enabled:

- ADC_AWDy_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADC_AWDy_OUT is reset after the end of the next guarded conversion that is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADC_AWDy_OUT is also reset when disabling the ADC (when setting ADDIS). Note that stopping regular or injected conversions (by setting ADSTP or JADSTP) has no influence on the generation of ADCx_AWDy_OUT.

Note: The AWDy flag is set by hardware and reset by software: the AWDy flag has no influence on the generation of ADC_AWDy_OUT (ex: ADC_AWDy_OUT can toggle while the AWDy flag remains at 1 if the software did not clear the flag).

The AWDy flag is cleared by programming it to 1 or by set ADDIS bit. ADSTP and JADSTP bits do not reset the AWDy flag.

Figure 327. ADC_AWDy_OUT signal generation (on all regular channels)

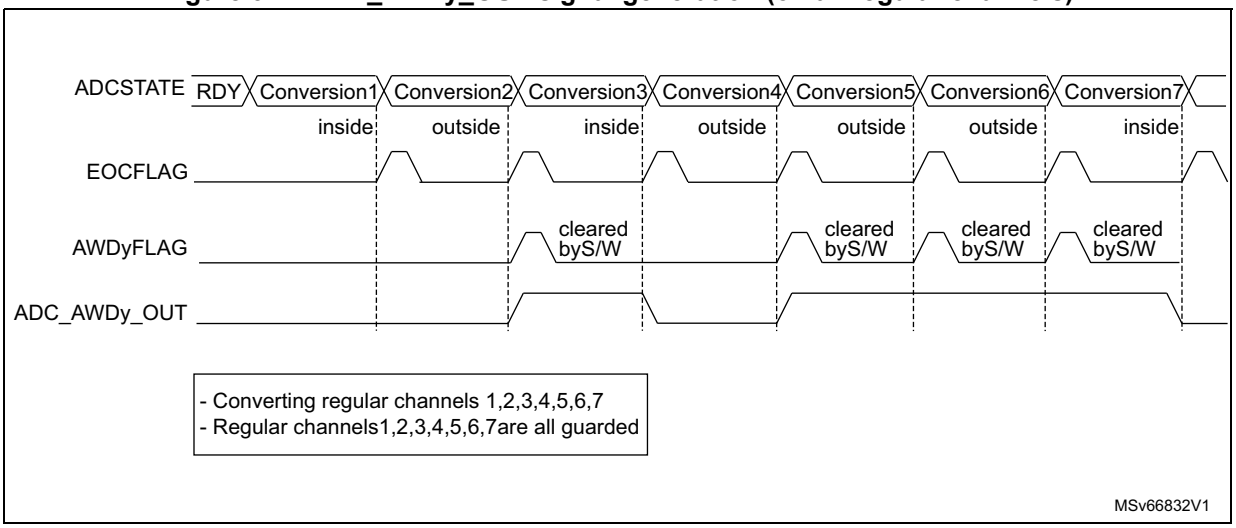


Figure 328. ADC_AWDy_OUT signal generation (AWDy flag not cleared by software)

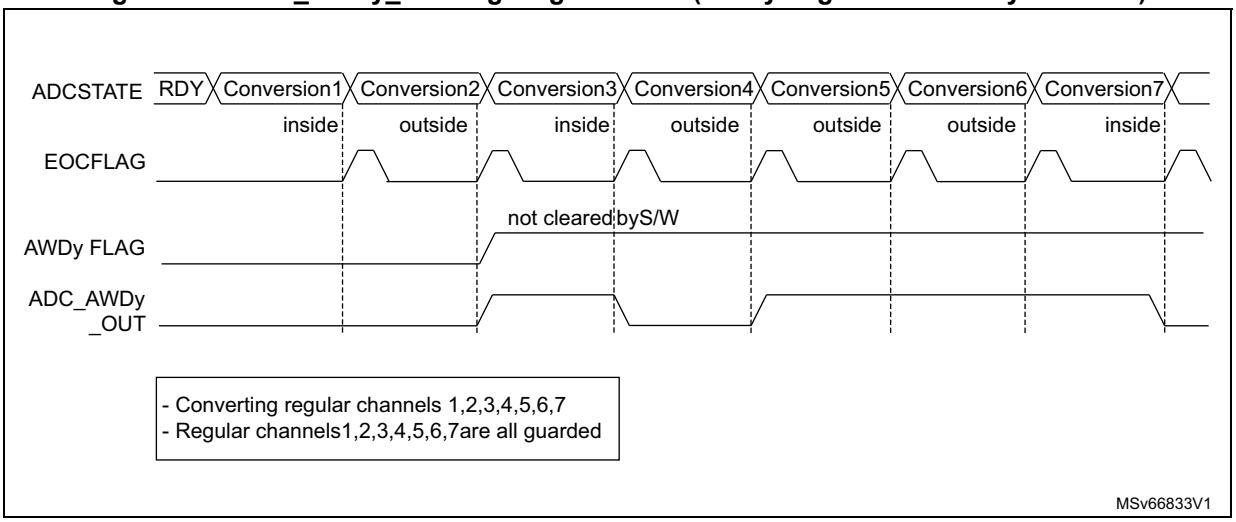


Figure 329. ADC_AWDy_OUT signal generation (on a single regular channel)

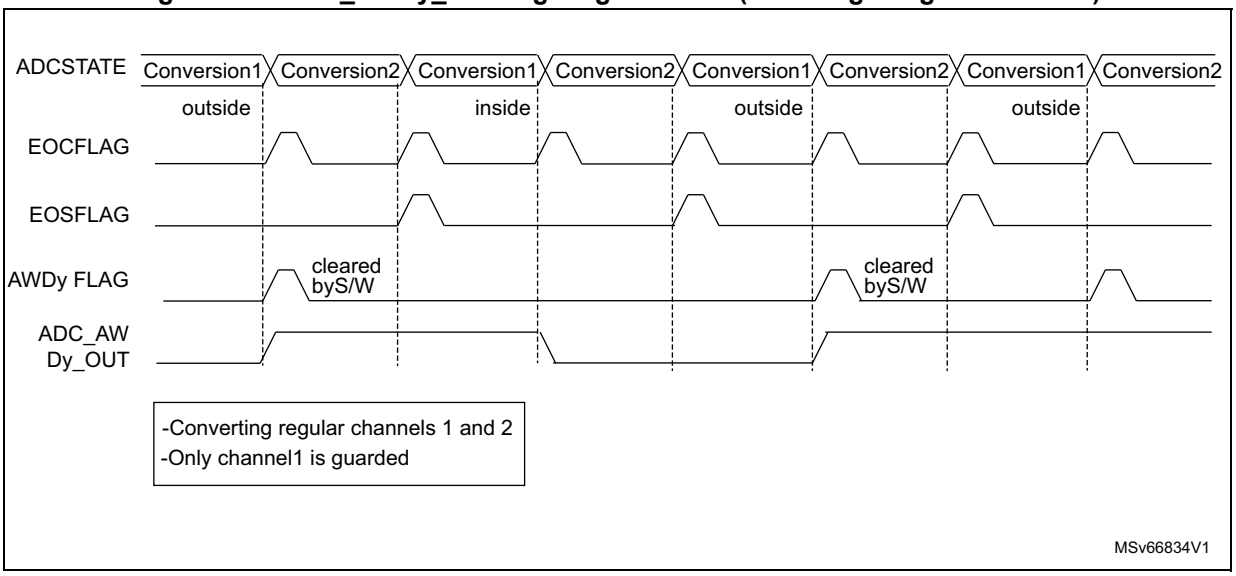
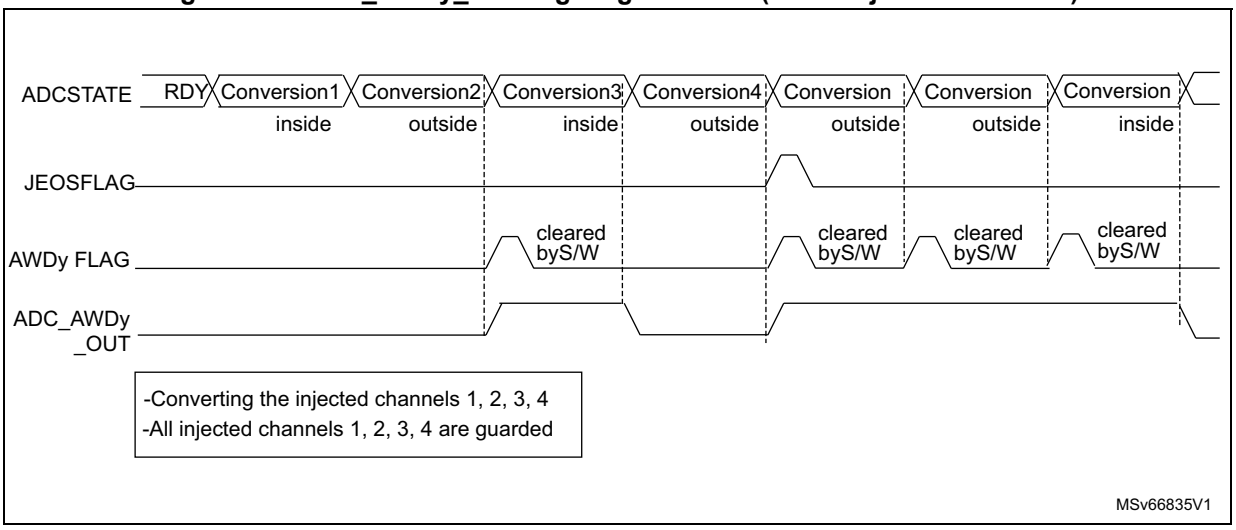


Figure 330. ADC_AWDy_OUT signal generation (on all injected channels)



Analog watchdog threshold control on the fly

LTR[22:0] and HTR[22:0] can be changed when an analog-to-digital conversion is ongoing (that is between the start of conversion and the end of conversion of the ADC internal state). If LTR[22:0] and HTR[22:0] are updated during the ADC conversion of the ADC guarded channel, resulting in analog watchdog thresholds to be applied from the next ADC conversion. The analog watchdog comparison is performed at each end of conversion. If the current ADC data is out of the new modified threshold, no interrupt and AWDy_OUT signal are issued. The Interrupt and the AWD generation only happen at the end of the conversion that started after the threshold update. If AWDy_OUT is already asserted, programming the new thresholds does not deassert the AWDy_OUT signal. However, the watchdog threshold must not be changed twice during the four adc_ker_ck periods.

Analog watchdog filter

With analog watchdog, a valid ADC conversion data range can be configured through the ADC_AWDxLTR and ADC_AWDxHTR registers. When the conversion results have consecutively passed the threshold for more than the watchdog filter value (AWDFILT[2:0] + 1), AWDx_OUT is generated as well as the AWDx flag. Before this value is reached, AWDx output is not activated. The analog watchdog filter can be used only when a single channel is selected for an AWD.

Analog watchdog with gain and offset compensation

When the gain and offset compensation are enabled, the analog watchdog compares the data after the compensation.

25.4.32 Oversampler

The oversampling unit performs data preprocessing to offload the device. It is able to handle multiple conversions and average them into a single data with increased data width, up to 22 bits.

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

It enables the following functions to be performed by hardware: averaging, data rate reduction, SNR improvement and basic filtering.

The oversampling ratio N is defined by the OVSR[9:0] bits in the ADC_CFGR2 register for regular conversion, or by the JOVSR[3:0] bits in the ADC_CFGR3 register for injected conversion. They can range from 2x to 1024x. The division coefficient *M* consists of a right bit shift up to 10 bits. It is defined through the OVSS[3:0] bits of the ADC_CFGR2 register, for regular conversion or JOVSS[3:0] bits of the ADC_CFGR3 for injected conversion.

The summation unit can yield a result up to 22 bits, which can be left or right shifted at the end. When the right shift is selected, it is rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred to the ADC_DR data register.

In this section, OVSS, LSHIFT, and OVSR are used to explain the functionality; however, the same descriptions also apply to JOVSS, JLSHIFT, and JOVSR for injected conversions.

Figure 331 gives a numerical example of the processing, from a raw 22-bit accumulated data to the final 12-bit result.

Figure 331. 12-bit result oversampling with 10-bit right shift and rounding

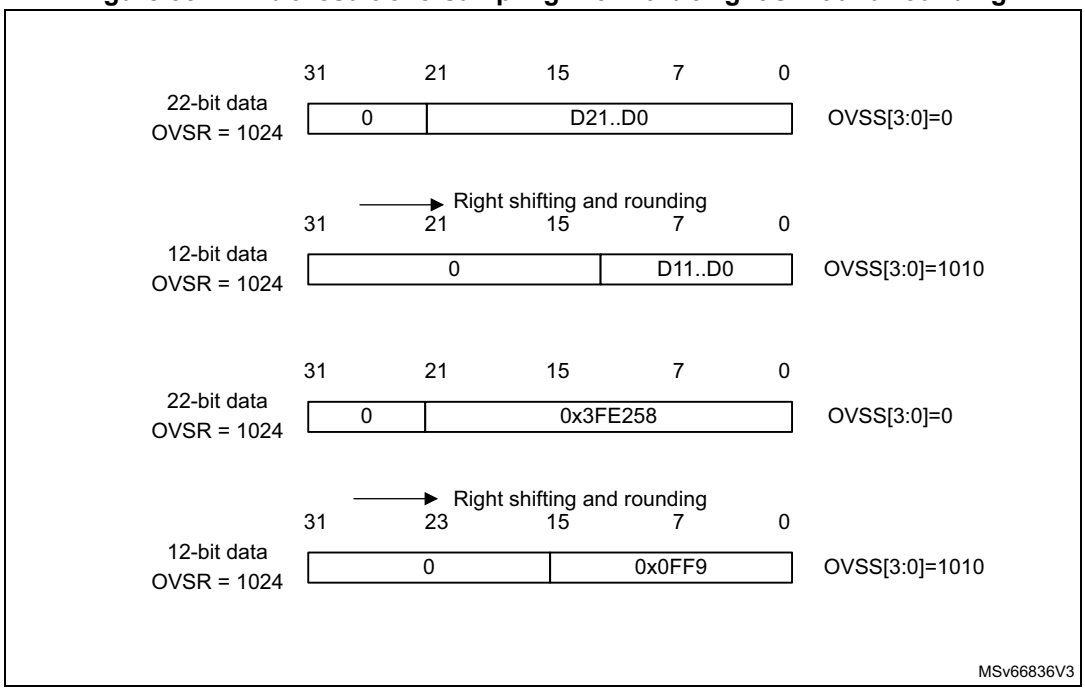


Table 319 gives data format for the various N and M combinations, for a raw conversion data equal to 0xFFF.

Table 251. Maximum output results versus N and M

Over sampling ratio	Max Raw data	No-shift OVSS = 0000	1-bit shift OVSS = 0001	2-bit shift OVSS = 0010	3-bit shift OVSS = 0011	4-bit shift OVSS = 0100	5-bit shift OVSS = 0101	6-bit shift OVSS = 0110	7-bit shift OVSS =0111	8-bit shift OVSS = 1000
x2	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x0020
x4	0x3FFC	0x3FFC	x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
x16	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
x64	0x3FFC0	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
x256	0xFFF00	0xFFF00	0x7FF80	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF
x1024	0x3FFC00	0x3FFC00	0x1FFE00	0xFFF00	0x7FF80	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC

The conversion timing rate does not change in oversampling mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every N conversion, with an equivalent delay equal to N x T_{CONV} = N x (t_{SMPL} + t_{SAR}). The flags are set as follows:

- The end of the sampling phase (EOSMP) is set after each sampling phase for regular conversion.
- The end of conversion (EOC) occurs once every N conversion, when the oversampled result is available.
- The end of sequence (EOS) occurs once the sequence of oversampled data is completed (that is after a total of N x sequence length conversions).

Unlike the conversion rate, the data latency changes in oversampling mode: each single conversion introduces an additional accumulation. As a result, an additional clock cycle is required when all the conversions are complete, after the accumulation and shift.
The EOC flag is set two additional clock cycles after the conversion is complete.

Analog watchdog

The analog watchdog functionality is maintained, with the following differences:

- The RES[1:0] bits are ignored and the comparison is always done using the full 23-bit values HTR[22:0] and LTR[22:0].
- The comparison is performed on the oversampled accumulated value before shifting.

Triggered mode

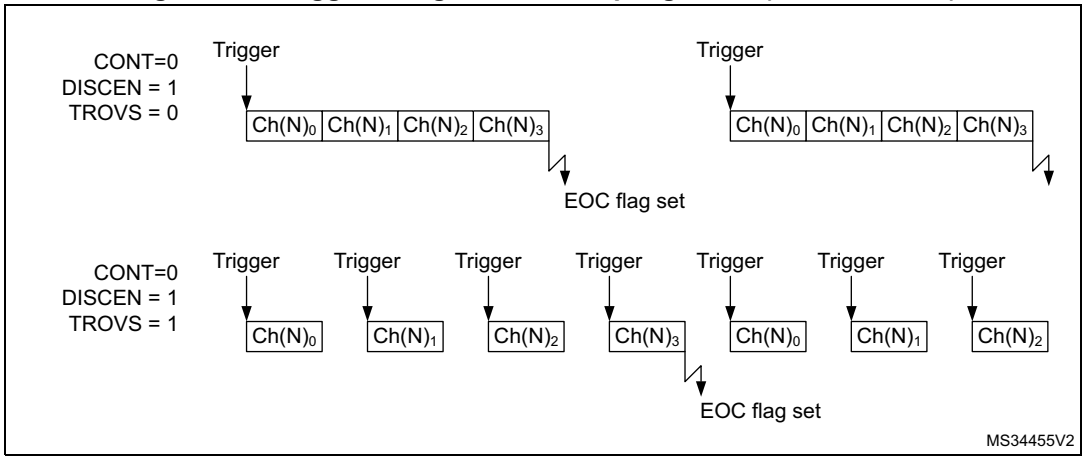
The oversampling can also be used for basic filtering purpose. Although it not a very powerful filter (slow roll-off and limited stop band attenuation), it can act as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched-mode power supply). For this purpose, a specific discontinuous mode can be enabled through the TROVS bit in ADC_CFGR2, to be able to achieve an oversampling frequency defined by the user and independent from the conversion time itself. The TROVS bit is supported with hardware trigger mode.

Figure 332 shows how conversions are started in response to triggers during discontinuous mode.

If the TROVS bit is set and OVSR ≠ 0, the content of the DISCEN bit is ignored and considered as 1.

If the TROVS bit is set, neither software trigger nor JAUTO is supported.

Figure 332. Triggered regular oversampling mode (TROVS bit = 1)



Injected and regular sequencer management in oversampling mode

In oversampling mode, injected and regular sequencers can have different behaviors. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously.

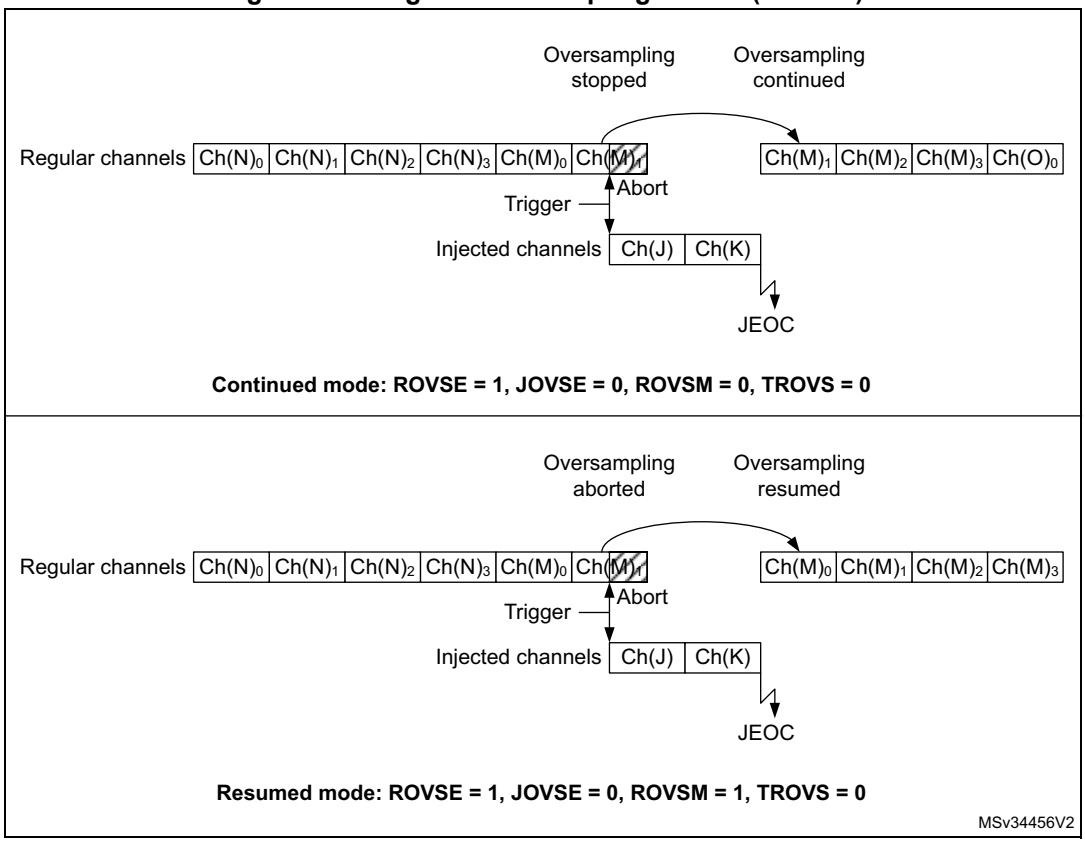
Oversampling regular channels only

The regular oversampling mode ROVSM bit defines how the regular oversampling sequence is resumed if it is interrupted by an injected conversion:

- In continuous mode (ROVSM = 0), the accumulation restarts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that oversampling is complete whatever the injection frequency (providing at least one regular conversion can be complete between triggers);
- In resumed mode (ROVSM = 1), the accumulation restarts from 0 (previous conversion results are ignored). This mode guarantees that all the data used for oversampling were converted back-to-back within a single time slot.

Figure 333 gives examples for a 4x oversampling ratio.

Figure 333. Regular oversampling modes (4x ratio)



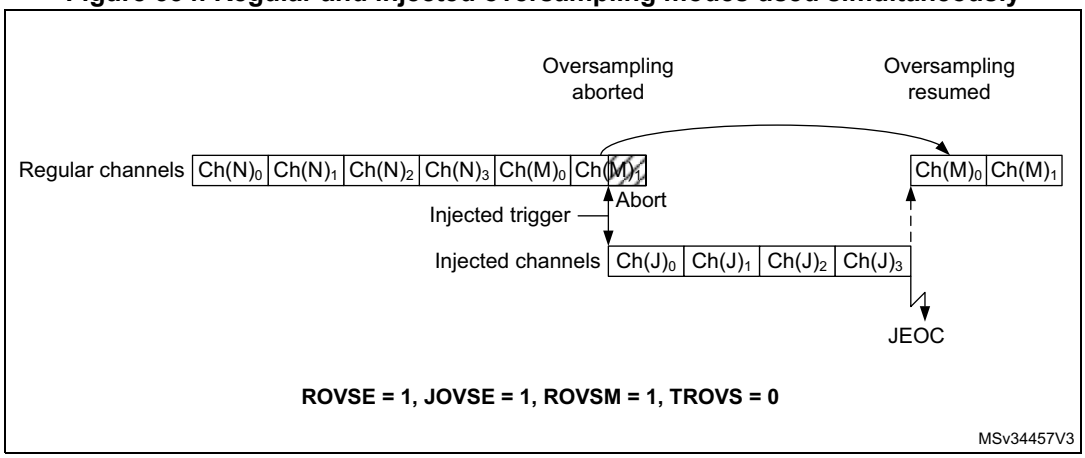
Oversampling injected channels only

The injected oversampling mode bit JOVSE enables oversampling solely for conversions in the injected sequencer.

Oversampling regular and injected channels

It is possible to have both ROVSE and JOVSE bits set. In this case, the regular oversampling must be resumed (ROVSM bit = 1), as shown in *Figure 334*. This limitation no longer applies in parallel injected oversampling mode.

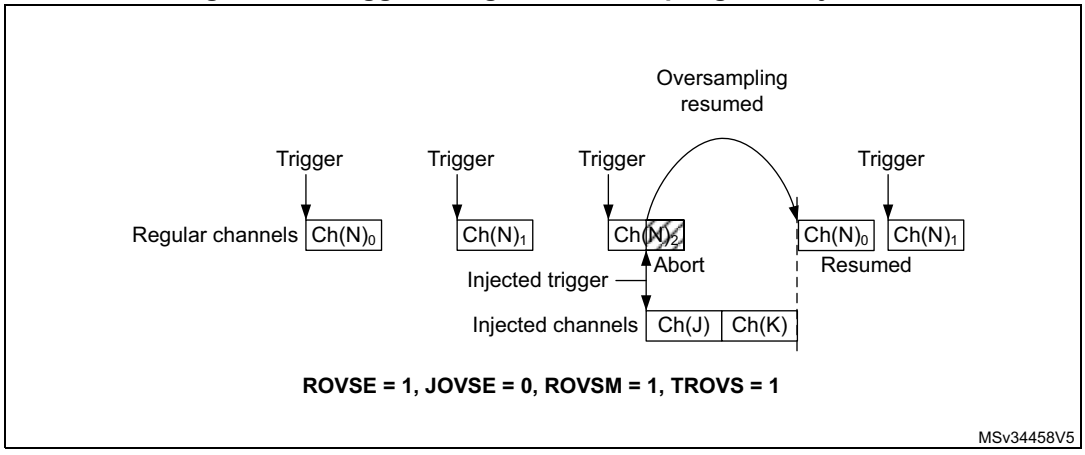
Figure 334. Regular and injected oversampling modes used simultaneously



Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected oversampling mode must be disabled and the ROVSM bit set (the resumed mode is forced). The JOVSE bit must be reset. The behavior is shown in *Figure 335*.

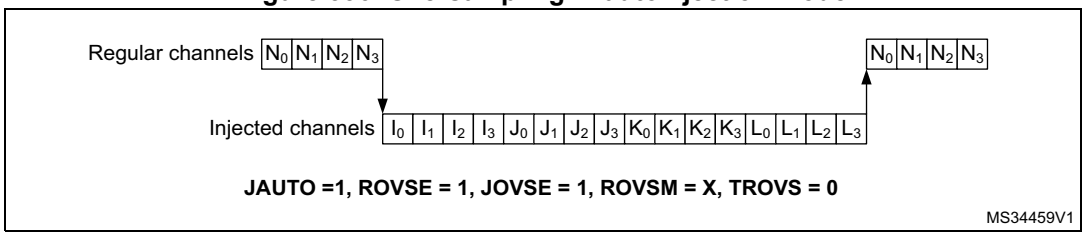
Figure 335. Triggered regular oversampling with injection



Autoinjection mode

It is possible to oversample autoinjected sequences and store all conversion results in registers. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1. Other combinations are not supported. The ROVSM bit is ignored in autoinjection mode. *Figure 336* shows how the conversions are sequenced.

Figure 336. Oversampling in autoinjection mode



Combined modes summary

Table 252 summarizes all combinations, including the modes that are not supported.

Table 252. Oversampler operating mode summary

Regular oversampling ROVSE	Injected oversampling JOVSE	Oversampler mode ROVSM 0 = continued 1 = resumed	Triggered regular mode TROVS	Comment
1	0	0	0	Regular continued mode
1	0	0	1	Not supported
1	0	1	0	Regular resumed mode
1	0	1	1	Triggered regular resumed mode
1	1	0	X	Not supported
1	1	1	0	Injected and regular resumed mode
1	1	1	1	Not supported
0	1	X	X	Injected oversampling

25.4.33 Parallel injected oversampling mode

The parallel injected oversampling mode is enabled by setting the JOVSPAR bit of the ADC_CFGR3 register.

To support parallel injected oversampling mode, four dedicated accumulators are implemented for injected conversions, one associated with each JDRy register.

This changes the injected and regular conversion sequences in oversampling mode described in *Section 25.4.32: Oversampler*, because the oversampling can be enabled simultaneously for all injected channels by setting the JOVSPAR bit.

For example, when the oversampling is enabled on two injected channels for scan conversion, the ADC performs the injected conversion (CH1[0], CH2[0], ... CH1[N-1], CH2[N-1]), then each channel is accumulated to ADC_JDR1 and ADC_JDR2 register.

The TJOVS bit of the ADC_CFGR3 register controls each individual conversion:

- When TJOVS is cleared, all conversions in the injected sequence are performed until the oversampling counter is reached.
- When TJOVS is set, each conversion requires an individual trigger.

TJOVS and TJOVSIVE can be used only when JOVSPAR = 1 and TJOVS is compatible with JDISCEN. When the JOVSIVE bit is set in the ADC_CFGR3 register, all intermediate accumulated values stored in the ADC_JDRy registers can be accessed.

When the JADSTP command is executed, the ADC_JDRy registers contain the intermediate accumulated data, and the JOVSCy[9:0] bitfield in the ADC_JOVSCRy register indicates how many accumulations have been performed.

The accumulation counter decrements from the selected ratio value minus one down to 0. Therefore, a value of 0 means that injected oversampling has completed the programmed number of accumulations. The initial counter value is 0, meaning no oversampling has been performed yet.

As long as the intermediate accumulated data count has not reached 0, the oversampling sequence is not complete, and the conversion data available in JDRy corresponds to the raw accumulated value without any right bit shift. The right bit shift is applied only once the oversampling sequence is fully completed.

If for any reason the injected oversampling needs to be stopped, the software can retrieve the intermediate data and the accumulation count, and calculate the data based on oversampling data divided by the accumulation counter.

Figure 337. Triggered injected parallel oversampling for 3 channels example

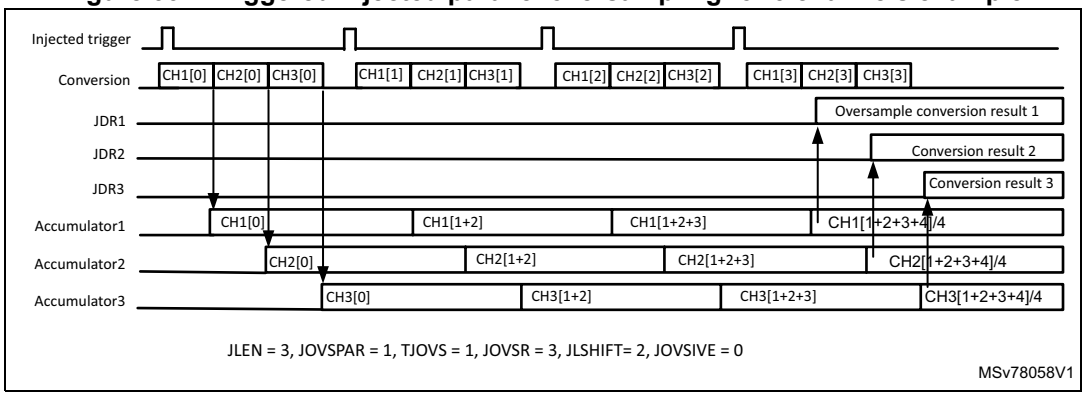


Figure 338. Injected oversampling with intermediate value enabled example

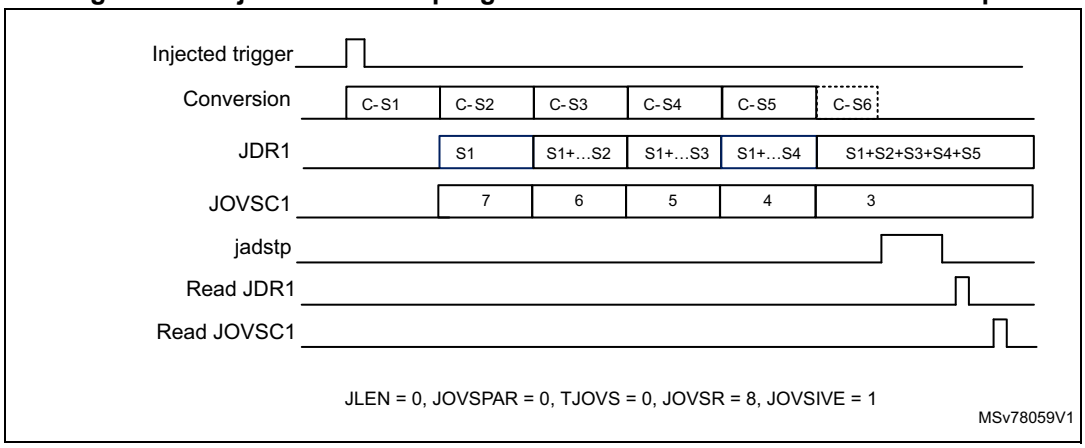
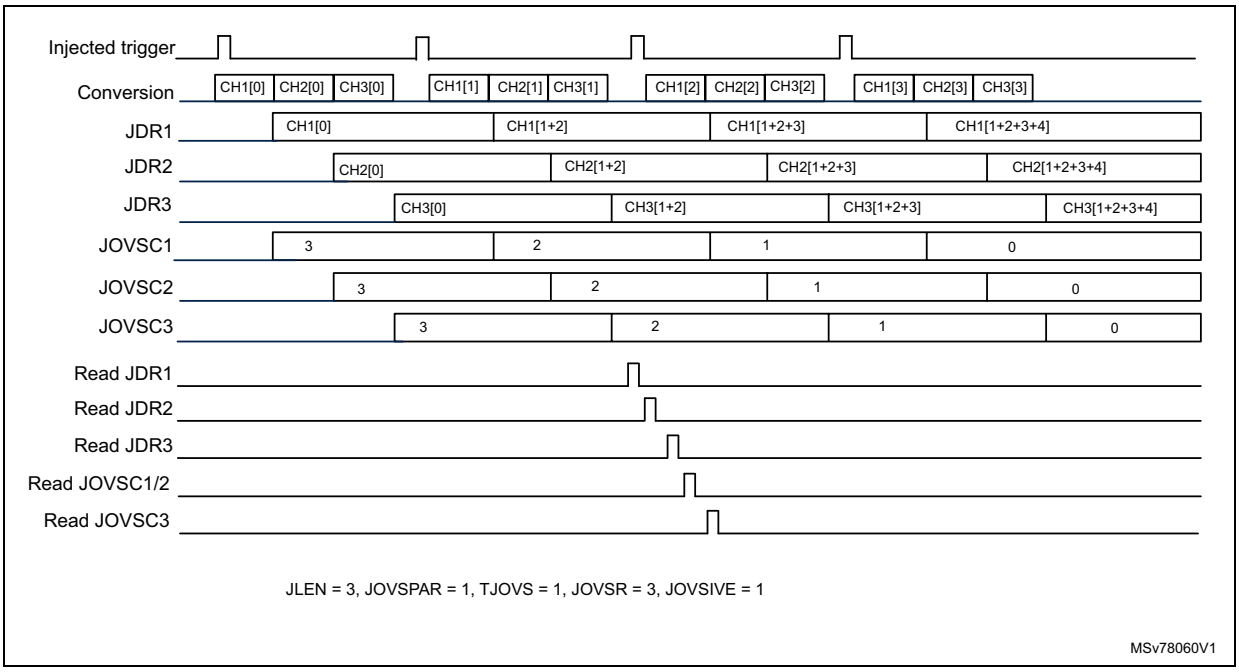


Figure 339. Triggered injected parallel oversampling for 3 channels with accumulated data example



25.4.34 Dual ADC modes

Dual ADC modes can be used in devices with two ADCs or more (see *Figure 340*).

In dual ADC mode the start of conversion is triggered alternately or simultaneously by the ADC master to the ADC slave, depending on the mode selected by the bits DUAL[4:0] in the ADCC_CCR register.

Four possible modes are implemented:

- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use these modes combined in the following ways:

- Injected simultaneous mode + regular simultaneous mode
- Regular simultaneous mode + alternate trigger mode
- Injected simultaneous mode + interleaved mode

In dual ADC mode (when the DUAL[4:0] bits in the ADCC_CCR register are not equal to zero), the following bits are shared between the master and slave ADCs, with the slave ADC bits being ignored and always applying to the corresponding bits of the master ADC:

- ADC_CFGR1 register: DMNGT[2:0], RES[1:0], EXTEN[1:0], OVRMOD, AUTDLY, DISCEN, DISCNUM[2:0], JDISCEN, JAUTO, JDISCEN.
- ADC_CFGR2 register: ROVSE, JOVSE, TROVS, ROVSM, BULB, SWTRIG, SMPTRIG, OVSR[9:0].
- ADC_GCOMP register: GCOMP.
- ADC_CR register: ADSTART, ADSTP, JADSTART, JADSTP.
- ADC_CFGR1 register: CONT, DISCEN, DISCNUM[2:0].
- ADC_SQR1 register: LEN (for the regular conversion).
- ADC_JSQR register: JLEN[1:0], JEXTSEL[4:0], JEXTEN[1:0] (for the injected conversion).
- ADC_CFGR3 register: JOVSPAR, JOVSE, JOVSIVE, TJOVS, JOVSR[9:0] (for the injected conversion).

For dual mode usage, the sampling time of the master and slave channels must be equal.

To start a conversion in dual mode, the user must program the EXTEN[1:0], EXTSEL, JEXTEN[1:0] and JEXTSEL bits of the master ADC only to configure a software or hardware trigger and a regular or injected trigger. The EXTEN[1:0], JEXTEN[1:0], and JEXTSEL bits of the slave ADC are don’t care).

In regular simultaneous or interleaved modes, once the user sets the ADSTART or ADSTP bit of the master ADC, the corresponding bit of the slave ADC is automatically set. However, ADSTART or ADSTP bit of the slave ADC is not necessarily cleared at the same time as the master ADC bit.

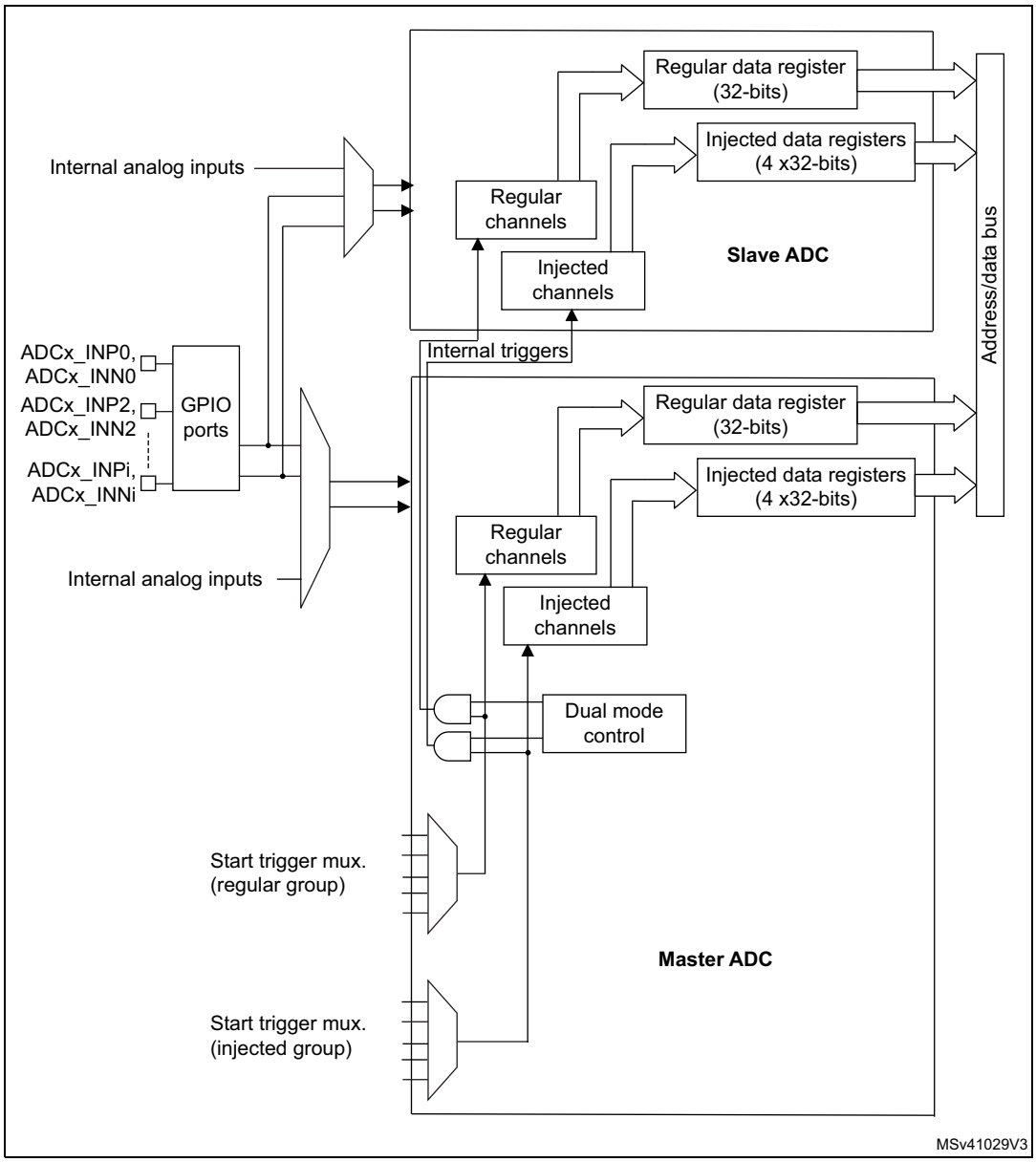
In injected simultaneous or alternate trigger modes, once the user sets the JADSTART or JADSTP bit of the master ADC, the corresponding bit of the slave ADC is automatically set. However, JADSTART or JADSTP bit of the slave ADC is not necessarily cleared at the same time as the master ADC bit.

In dual ADC mode, the converted data of the master and slave ADC can be read in parallel, by reading the ADC common data register (ADCC_CDR). The flags can also be read in parallel by reading the dual-mode status register (ADCC_CSR).

Note: In dual mode, BULB, JAUTO, SMPTRIG, and the injected queue are not supported.

During ADC calibration, the DUAL[4:0] bits must be set to 00000.

Figure 340. Dual ADC block diagram



1. External triggers also exist on slave ADC but are not shown for the purposes of this diagram.
2. The ADC common data register (ADCC_CDR) contains both the master and slave ADC regular converted data.

Injected simultaneous mode with independent regular conversion

The injected simultaneous mode is selected by programming DUAL[4:0] bits to 0b00101.

This mode converts an injected group of channels. The external trigger source comes from the injected group multiplexer of the master ADC (selected by the JEXTSEL bits in the ADC_JSQR register).

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

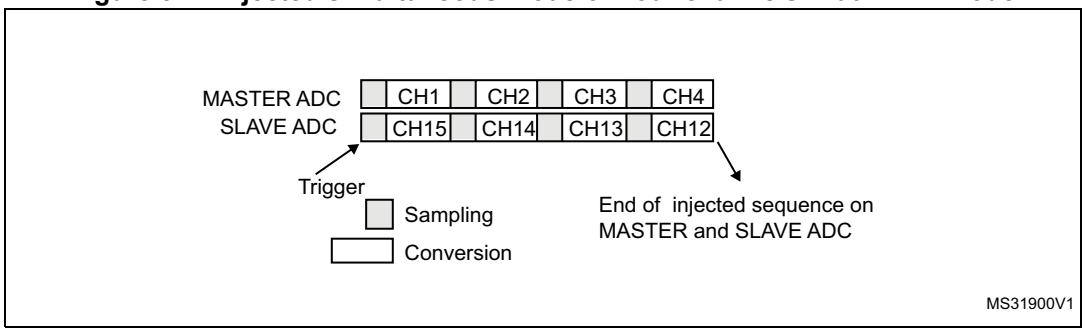
In simultaneous mode, the user software must convert sequences with the same length. Sampling times for channels with the same sequence number must be equal.

Regular conversions can be performed on one or all ADCs. In that case, they are independent of each other and are interrupted when an injected event occurs. They are resumed at the end of the injected conversion group.

The software is notified by interrupts when it can read the data:

- If the duration of the master-injected sequence is equal to the duration of the slave injected one (as shown in *Figure 341*), the software can enable only one of the two JEOS interrupts (for example the master JEOS) and read both converted data (from master ADC_JDRy and slave ADC_JDRy registers).

Figure 341. Injected simultaneous mode on four channels: Dual ADC mode



If JDISCEN is set, each simultaneous conversion of the injected sequence requires an injected trigger event to occur.

This mode can be combined with the AUTDLY mode:

- Once a simultaneous injected sequence of conversions has ended, a new injected trigger event is accepted only if both JEOS bits of the master and the slave ADC have been cleared (delay phase). Any new injected trigger events occurring during the ongoing injected sequence and the associated delay phase are ignored.
- Once a regular sequence of conversions of the master ADC has ended, a new regular trigger event of the master ADC is accepted only if the master data register (ADC_DR) has been read. Any new regular trigger events occurring for the master ADC during the ongoing regular sequence and the associated delay phases are ignored.
  There is the same behavior for regular sequences occurring on the slave ADC.

In this mode, the following register bits are shared between the master and slave ADCs:

JADSTART, JADSTP, JDISCEN, JOVSE, JLEN, JEXTEN, JEXTSEL, and GCOMPEN, JOVSPAR, JOVSIVE, JOVSE, and TJOVS.

Regular simultaneous mode with independent injected conversions

The regular simultaneous mode is selected by programming DUAL[4:0] bits to 0b00110.

This mode applies to a regular group of channels. The external trigger source comes from the regular group multiplexer of the master ADC (selected by the EXTSEL bits in the ADC_CFGR1 register). A simultaneous trigger is provided to the slave ADC.

In this mode, independent injected conversions are supported. An injection request (either on the master or on the slave) aborts the current simultaneous conversions both for master and slave. They are restarted once the injected conversion is completed.

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In regular simultaneous mode, the user software must convert sequences with the same length. Sampling times for channels with the same sequence number must be equal.

The software is notified by interrupts when it can read the data:

- As the duration of the master regular sequence is equal to the duration of the slave one (as shown in *Figure 342*), the software can enable only one of the two EOC interrupts (for example the master EOC) and read both converted data from the common data register (ADCC_CDR).
  The user can read data from either ADCC_CDR or ADCC_CDR2. Once software selects one of these registers, it must continue using the same register until the ADSTP procedure is executed.

The regular data can also be read using the DMA. Several methods are possible:

- Use two DMA channels (one for the master and one for the slave). In this case DAMDF[1:0] bits must be kept cleared:
  a) Configure the DMA master ADC channel to read ADC_DR from the master. DMA requests are generated at each EOC event of the master ADC.
  b) Configure the DMA slave ADC channel to read ADC_DR from the slave. DMA requests are generated at each EOC event of the slave ADC.
- Configure the dual ADC mode data format DAMDF[1:0] bits, which leaves one DMA channel free for other uses:
  a) Set DAMDF[1:0] = 0b10 or 0b11 (depending on resolution).
  b) A single DMA channel is used (the one corresponding to the master). Configure the DMA master ADC channel to read the common ADC data register (ADCC_CDR or ADCC_CDR2).
  c) A single DMA request is generated each time both the master and slave EOC events occur. At that moment:
  – The slave ADC converted data is available in the upper half-word of the 32-bit ADCC_CDR register
  – The master ADC converted data is available in the lower half-word of the ADCC_CDR register.

Alternatively, data can be read from ADCC_CDR2:

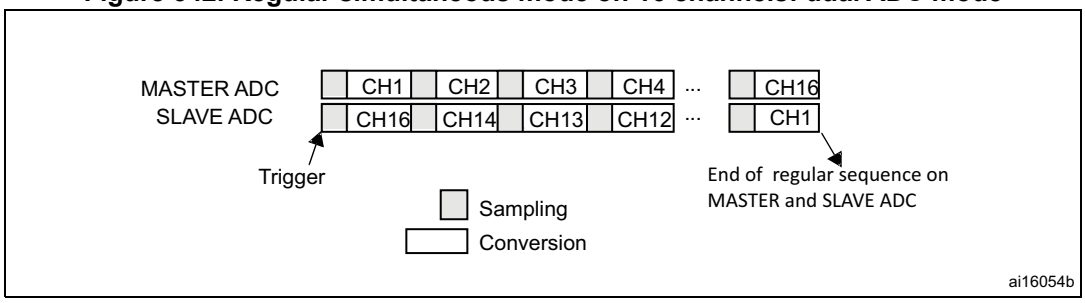
– The first read returns the master ADC data.

– The second read returns the slave ADC data.
After these reads, a DMA request is generated. The DMA transfers the slave data from the same data register and handles two transfers for dual mode operation.

d) Both EOC flags are cleared when the DMA reads the ADCC_CDR register.

Note: When DAMDF[1:0] = 0b10 or 0b11, the user must program the same number of conversions in the master and in the slave sequence. Otherwise, the remaining conversions do not generate a DMA request.

Figure 342. Regular simultaneous mode on 16 channels: dual ADC mode



If the DISCEN bit is set, each "*n*" simultaneous conversion of the regular sequence require a regular trigger event to occur ("n" is defined by DISCNUM).

This mode can be combined with the AUTDLY mode:

- Once a simultaneous conversion of the sequence has completed, the next conversion in the sequence starts only after the EOC flag of both the master and the slave are cleared.
- Once a simultaneous regular sequence of conversions has completed, a new regular trigger event is accepted only after the EOC flag of both the master and the slave are cleared. Any new regular trigger events occurring during the ongoing regular sequence and the associated delay phases are ignored.

The DMA can be used to handle data in regular simultaneous mode combined with AUTDLY mode, assuming that multi-DMA mode is used: bits DAMDF must be set to 0b10 or 0b11.

Interleaved mode with independent injected conversions

This mode is selected by programming DUAL[4:0] bits to 0b00111.

It can be started only on a regular group (usually one channel). The external trigger source comes from the regular channel multiplexer of the master ADC.

After an external trigger occurs:

- The master ADC starts immediately,
- The slave ADC starts after a delay of several ADC clock cycles after the sampling phase of the master ADC has completed (during the master ADC conversion period).

The minimum delay between two conversions in interleaved mode is configured in the DELAY bits in the ADCC_CCR register (see *Table DELAY bits versus ADC resolution*). This delay starts counting one half cycle after the end of the sampling phase of the master

conversion. This way, an ADC cannot start a conversion if the complementary ADC is still sampling its input (only one ADC can sample the input signal at a given time).

- The minimum possible DELAY is 1 to ensure that there is at least one cycle time between the opening of the analog switch of the master ADC sampling phase and the closing of the analog switch of the slave ADC sampling phase.
- The maximum DELAY is equal to the number of cycles corresponding to the selected resolution. However, the user must properly calculate this delay to ensure that an ADC does not start a conversion while the other ADC is still sampling its input.

As the CONT bit is set on both the master and slave ADCs, the selected regular channels of both ADCs are continuously converted.

The software is notified by interrupts when it can read the data at the end of each conversion event (EOC) on the master or slave ADC. A slave and master EOC interrupts are generated (if EOCIE is enabled) and the software can read the ADC_DR of the slave/master ADC.

Table 253. DELAY bits versus ADC resolution

DELAY bits	12-bit resolution	10-bit resolution	8-bit resolution	6-bit resolution
0000	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$
0001	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$
0010	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$
0011	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$
0100	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$
0101	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
0110	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
0111	8 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
1000	9 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
1001	10 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
1010	11 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
1011	12 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
1100	13 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
others	13 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$

Figure 343. Interleaved mode on 1 channel in continuous conversion mode: Dual ADC mode

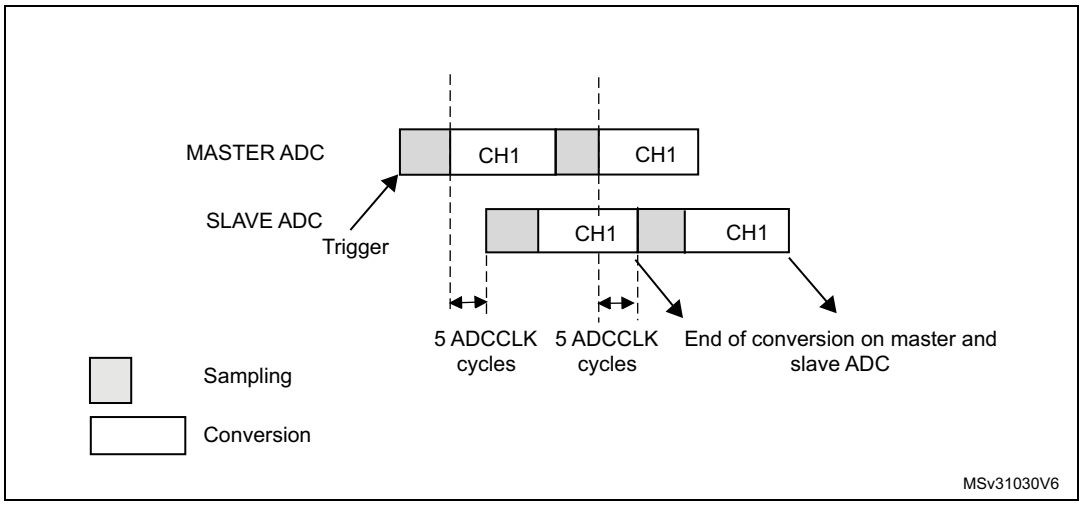
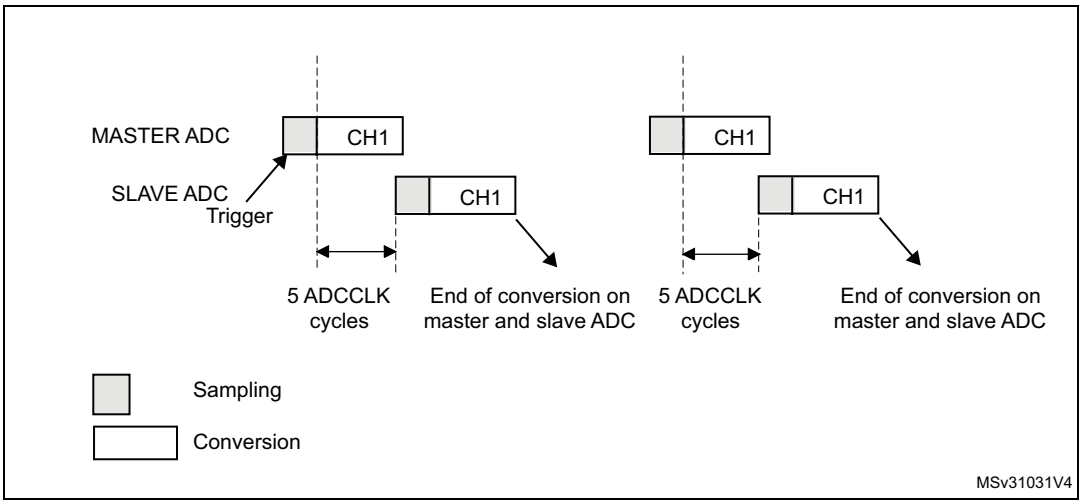


Figure 344. Interleaved mode on 1 channel in single conversion mode: Dual ADC mode



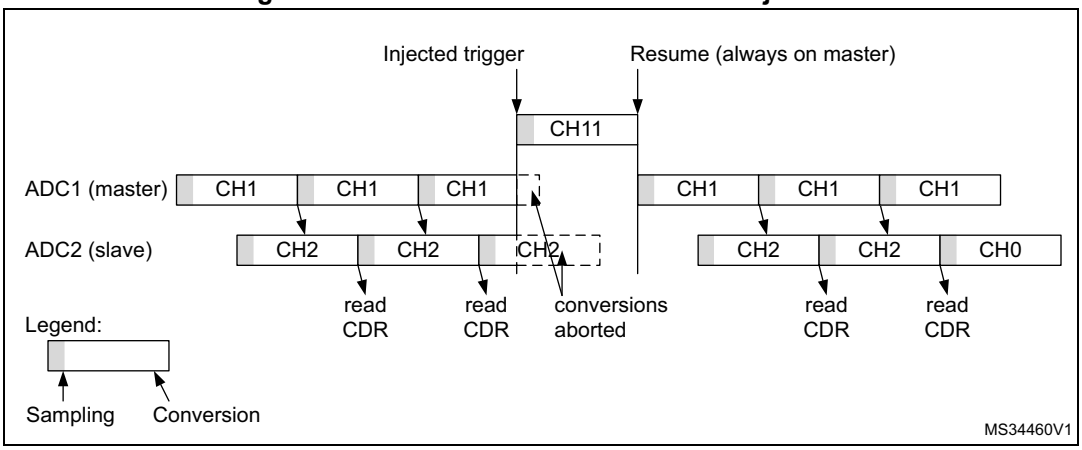
If the DISCEN bit is set, each "*n*" simultaneous conversion ("*n*" is defined by DISCNUM) of the regular sequence require a regular trigger event to occur.

In interleaved mode, injected conversions are supported. When injection is done (either on master or slave), both master and the slave regular conversions are aborted and the sequence is restarted from the master (see *Figure 345*).

Note: In interleaved mode, the ADCx_AWDy_OUT signal for the master ADC is synchronized with the EOC of the master ADC, but the interrupt flag is synchronized with the EOC/JEOC flag of the slave ADC.

In addition, the ROVSM = 0 setting is not supported in oversampling mode.

Figure 345. Interleaved conversion with injection



Alternate trigger mode with independent regular conversions

The alternate trigger mode is selected by programming DUAL[4:0] bits to 0b01001.

This mode can only be started on an injected group. The source of the external trigger comes from the multiplexer injected group of the master ADC.

This mode is only possible when selecting hardware triggers: JEXTEN[1:0] must not be 0x0.

Injected discontinuous mode disabled (JDISCEN = 0)

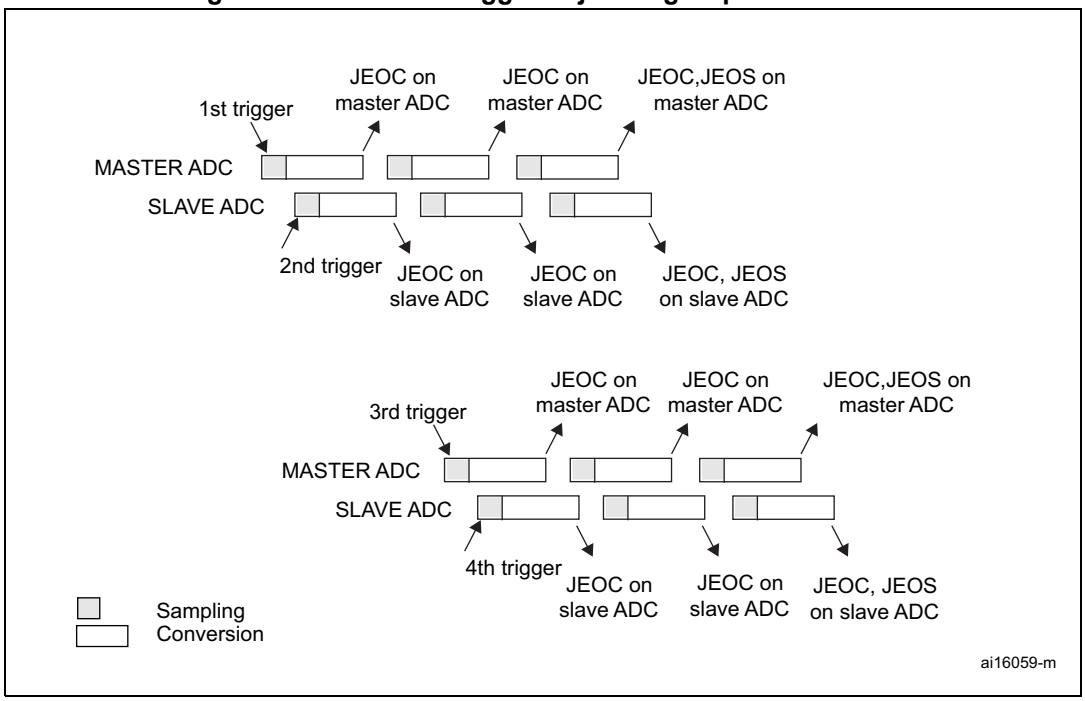
1. When the first trigger occurs, all injected master ADC channels in the group are converted.
2. When the second trigger occurs, all injected slave ADC channels in the group are converted.
3. And so on...

A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

Figure 346. Alternate trigger: injected group of each ADC



Note: Regular conversions can be enabled on one or all ADCs. In this case, the regular conversions are independent of each other. A regular conversion is interrupted when the ADC has to perform an injected conversion. It is resumed when the injected conversion is finished.

The time interval between two trigger events must be greater than or equal to one ADC clock period. The minimum time interval between two trigger events that start conversions on the same ADC is the same as in the single ADC mode.

Injected discontinuous mode enabled (JDISCEN = 1)

If the injected discontinuous mode is enabled for both master and slave ADCs:

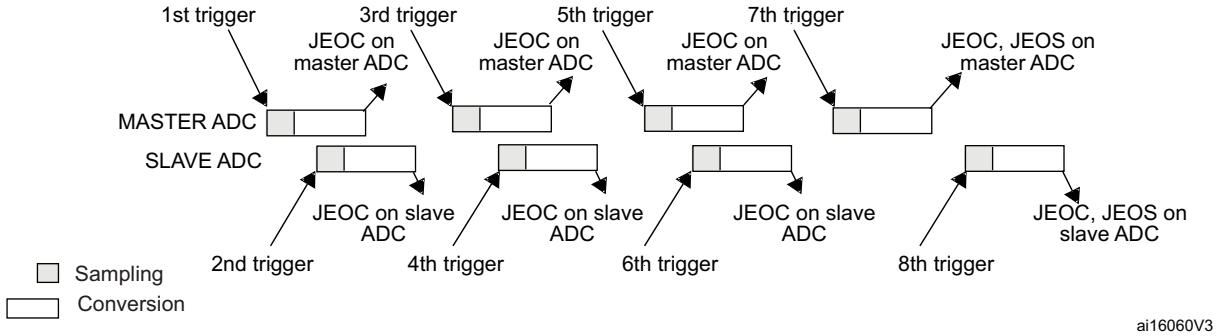
1. When the first trigger occurs, the first injected channel of the master ADC is converted.
2. When the second trigger occurs, the first injected channel of the slave ADC is converted.
3. And so on...

A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

Figure 347. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode



Combined regular/injected simultaneous mode

This mode is selected by programming DUAL[4:0] bits to 0b00001.

The simultaneous conversion of a regular group can be interrupted to start the simultaneous conversion of an injected group.

Note: In combined regular/injected simultaneous mode, the user software must convert sequences with the same length.

Combined regular simultaneous and alternate trigger mode

This mode is selected by programming DUAL[4:0] bits to 0b00010.

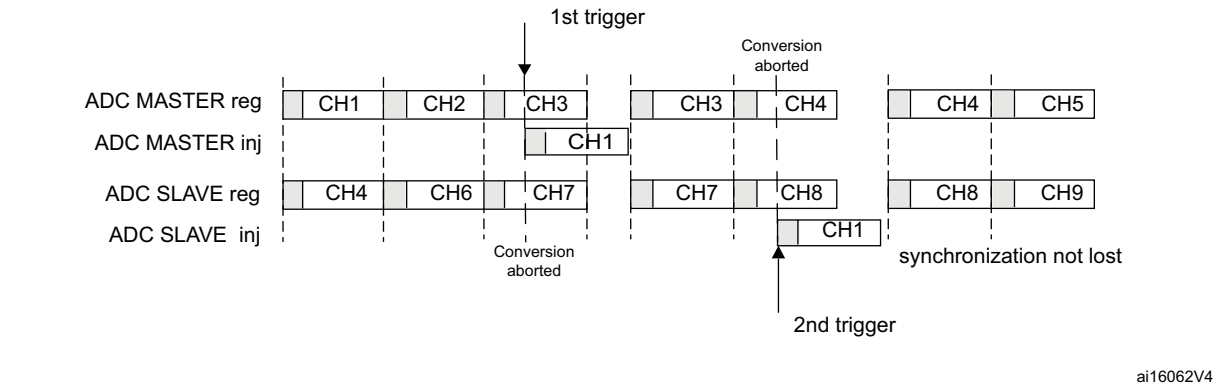
The simultaneous conversion of a regular group can be interrupted to start the alternate trigger conversion of an injected group. *Figure 348* shows the behavior of an alternate trigger interrupting a simultaneous regular conversion.

The injected alternate conversion immediately starts after the injected event. If a regular conversion is already running, in order to ensure synchronization after the injected conversion, the regular conversion of all (master/slave) ADCs is stopped and resumed synchronously at the end of the injected conversion.

Note: In combined regular simultaneous + alternate trigger mode, the user software must convert sequences with the same length.

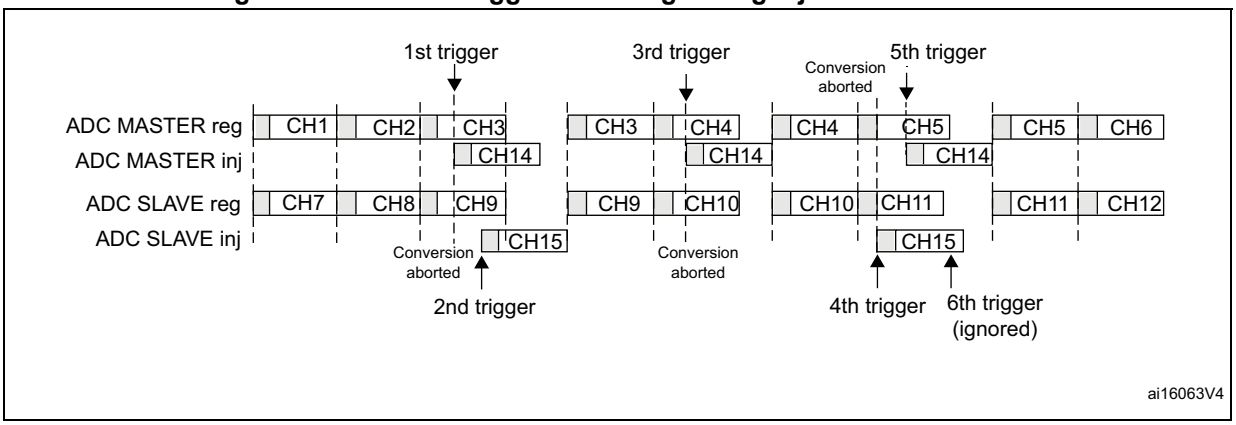
The software trigger is not supported for injected conversions.

Figure 348. Alternate + regular simultaneous



If a trigger occurs during an injected conversion that has interrupted a regular conversion, the alternate trigger is served. *Figure 349* shows the behavior in this case (note that the sixth trigger is ignored because the associated alternate conversion is not complete).

Figure 349. Case of trigger occurring during injected conversion



Combined injected simultaneous and interleaved mode

This mode is selected by programming DUAL[4:0] bits to 0b00011.

An interleaved conversion can be interrupted with a simultaneous injected event.

In this case, the interleaved conversion is immediately interrupted and the simultaneous injected conversion starts. At the end of the injected sequence, the interleaved conversion is resumed. When the interleaved regular conversion resumes, the first regular conversion which is performed is always the master's one. *Figure 350*, *Figure 351*, and *Figure 352* show examples of this behavior.

Note: In this mode, the ROVSM = 0 setting is not supported in oversampling mode.

Figure 350. Interleaved single channel CH0 with injected sequence CH11, CH12

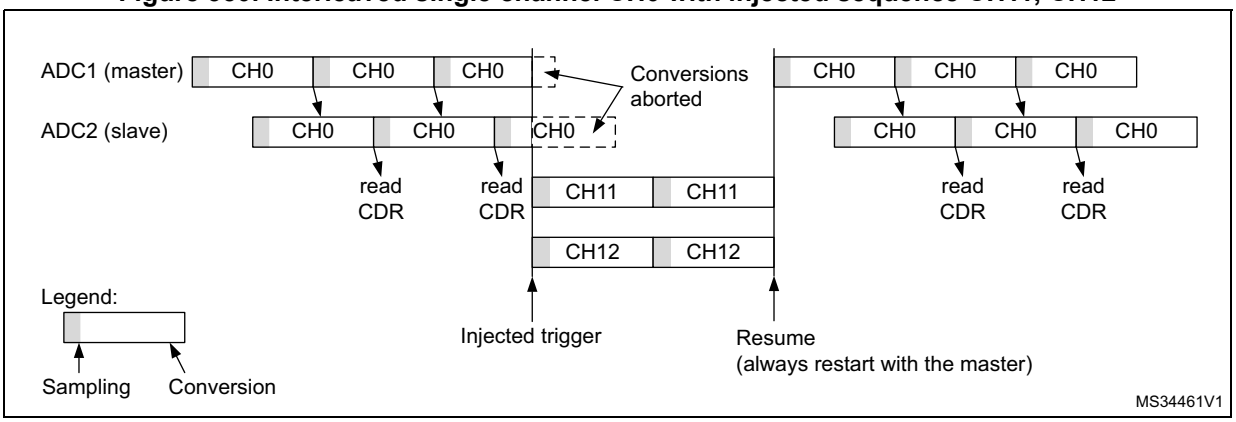


Figure 351. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 case 1: Master interrupted first

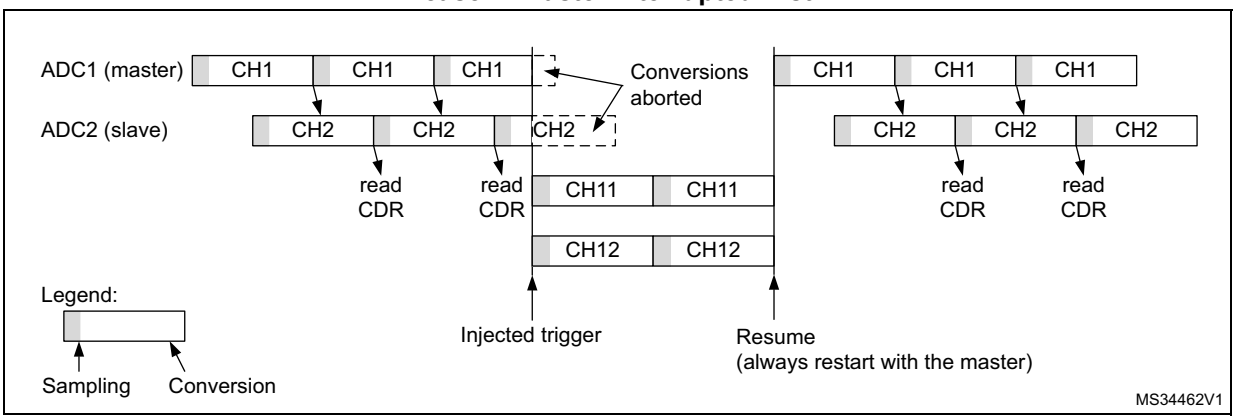
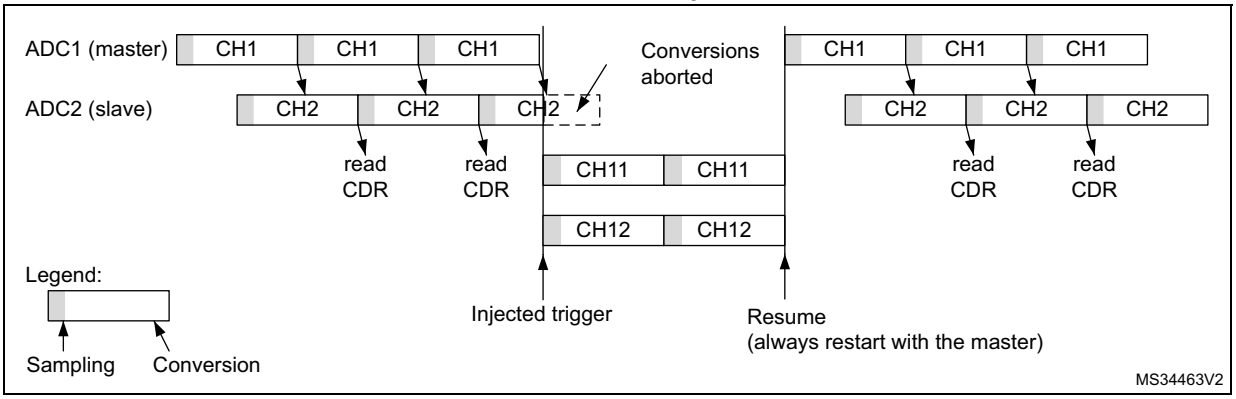


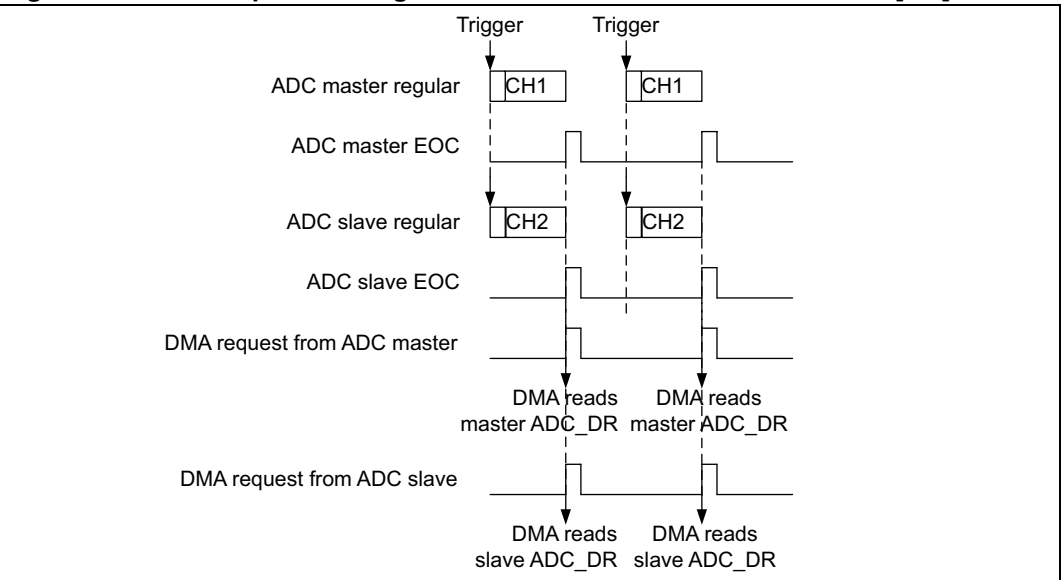
Figure 352. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 case 2: Slave interrupted first



DMA requests in dual ADC mode

In all dual ADC modes, it is possible to use two DMA channels (one for the master, one for the slave) to transfer the data, like in single mode (refer to *Figure 353: DMA requests in regular simultaneous mode when DAMDF[1:0] = 0b00*).

Figure 353. DMA requests in regular simultaneous mode when DAMDF[1:0] = 0b00



Configuration where each sequence contains only one conversion. Master and slave timings are equal.

MSv31032V3

In interleaved mode or in simultaneous regular mode, it is also possible to save one DMA channel and transfer both data using a single DMA channel. To do this, the DAMDF[1:0] bits must be configured in the ADCC_CCR register:

- **DAMDF[1:0] = 0b10, 32-bit format:**

  A DMA request is generated alternatively after the master and slave EOC events have occurred. At that time, the data are alternatively available in the 32-bit register ADCC_CDR2 32-bit register.

  This mode is used when the data width is above 16 bits (conversion data width can exceed 16 bits when functions such as oversampling, compensation and left-bit shift, are used).

  Behavior: A DMA request is generated each time a new 32-bit data is available:

  – First DMA request: ADCC_CDR2[31:0] = MST_ADC_DR[31:0]

  – Second DMA request: ADCC_CDR2[31:0] = SLV_ADC_DR[31:0]

- **DAMDF[1:0] = 0b10, 16-bit format**:

  A DMA request is generated alternatively after the master and slave EOC events have occurred. At that time, two data items are available and the 32-bit register ADCC_CDR contains the two half-words representing two ADC-converted data. The slave ADC data takes the upper half-word and the master ADC data takes the lower half-word. DMA transfers two data by one request.

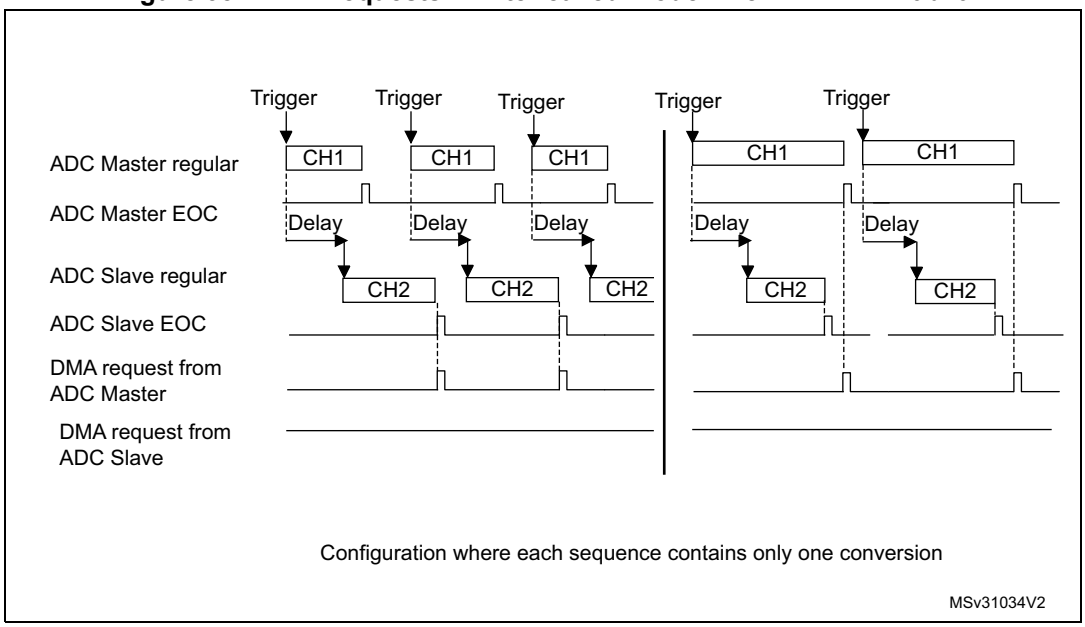
Any value above 16-bit in the master or the slave-converted data is truncated to the least 16 significant bits.

Behavior: A DMA request is generated each time a new 32-bit data is available:

– First DMA request:
ADCC_CDR[31:0] = (SLV_ADC_DR[15:0] << 16) | MST_ADC_DR[15:0]

– Second DMA request:
ADCC_CDR[31:0] = SLV_ADC_DR[15:0] | MST_ADC_DR[15:0]

Figure 354. DMA requests in interleaved mode when DAMDF = 0b10



Note: When using dual ADC mode, the user must take care to configure properly the duration of the master and slave conversions so that a DMA request is generated and served for reading both data (master + slave) before a new conversion data is available.

- **DAMDF[1:0] = 0b11**:

  This mode is similar to the DAMDF[1:0] = 0b10 configuration. The only difference is that on each DMA request (two data are available), two bytes representing two ADC converted data are transferred as a half-word.

  This mode is used when the resolution is 6 bits or 8 bits and data are not signed.

  Behavior: A DMA request is generated each time two data items are available:

  – First DMA request:
  ADCC_CDR[15:0] = (SLV_ADC_DR[7:0] << 8)| MST_ADC_DR[7:0]

  – Second DMA request:
  ADCC_CDR[15:0] = SLV_ADC_DR[7:0] | MST_ADC_DR[7:0]

Note: DMA burst mode is not supported in ADC dual mode.

Overrun detection when OVRMOD = 0

In dual ADC mode (when DUAL[4:0] is not equal to b00000), if an overrun is detected on one of the ADCs, DMA requests are no longer issued to ensure that all the data transferred to the RAM are valid (this behavior occurs whatever the DAMDF configuration). The EOC bit

corresponding to one ADC might remains set because the data register of this ADC contains valid data.

Overrun detection when OVRMOD = 1

In dual ADC mode, if an overrun occurs on either ADC, the data register is overwritten, causing continuous DMA requests. This results in loss of data coherency.

DMA one-shot mode/ DMA circular mode when multi-ADC mode is selected

When DAMDF mode is selected (0b10 or 0b11), DMNGT[1:0] bits of the ADCC_CCR register must also be configured to select between DMA one-shot mode and circular mode, as explained in section *Section : Managing conversions using the DMA*

Stopping the conversions in dual ADC modes

In dual ADC mode, the user software must set the ADSTP/JADSTP control bits in the master ADC to stop the conversions of both ADC. The other ADSTP control bit on the slave ADC has no effect in dual ADC mode.

Once both ADCs are effectively stopped, the ADSTART/JADSTART bits on the master and slave ADCs are both deasserted by hardware.

Analog watchdog in dual mode

The analog watchdog is supported in all dual modes.

Disabling the ADC in dual mode

During a dual-mode operation, the master ADC shares the ADC clock with the slave ADC. To disable the ADC correctly through the ADDIS bit of ADC_CR, set the ADSTP (or JADSP) bit on the master ADC. Once ADSTP (or JADSTP) becomes 0 both for the master and slave ADCs, set the ADDIS bit on the master ADC, then set the ADDIS bit on the slave ADC.

Autoinjection mode in dual mode

Autoinjection mode is not supported when DUAL[4:0] equals 0b00010, 0b00011, 0b00101, 0b00110, 0b00111, or 0b01001.

Additionally, parallel injected oversampling mode is not supported in autoinjection mode.

ADF mode in dual ADC interleaved mode

In dual ADC interleaved modes (DUAL[4:0] = 0b00011 or DUAL[4:0] = 0b00111), the ADC conversion results can be transferred directly to the ADF.

This mode is enabled by setting DMNGT[1:0] bits to 0b10 in the master ADC ADC_CFGR1 register.

The ADC transfers alternatively the 16 least significant bits of the regular data register from the master and the slave converter to a single channel of the ADF. Each transfer resets the EOC flag of each channel once the transfer is complete.

The data format must be 16-bit signed:

ADC_DR[15:12] = Sign extended

ADC_DR[11] = Sign

ADC_DR[10:0] = Data

To obtain 16-bit signed format, the software must configure OFFSET[11:0] bits to 0x800 with OFFSETy_CH[4:0] selecting the target channel set.

Only right aligned data can be set as ADF input format (see *Figure 316: Right alignment (offset enabled, signed value)*).

ADF mode in dual ADC simultaneous mode

It is not mandatory to use the ADF in dual ADC simultaneous mode since conversion data are handled by each individual channel. Single mode with same trigger source results in simultaneous conversion with ADF interface.

Dual ADC modes supported in oversampling mode

It is possible to have oversampling enabled when working in dual ADC configuration. In this case, the two ADCs must be programmed with the very same settings (including oversampling).

All other dual ADC modes are not supported when either regular or injected oversampling is enabled (that means when ROVSE = 1 or JOVSE = 1).

25.4.35 Temperature sensor

The temperature sensor can be used to measure the junction temperature (T_J) of the device.

The temperature sensor is internally connected to the ADC input channels, which are used to convert the sensor output voltage to a digital value (see *Table: ADC interconnection* in *Section 25.4.2: ADC pins and internal signals* for more details). When not in use, the sensor can be put in power-down mode. It supports the temperature range specified in the product datasheet.

Figure 355 shows the block diagram of connections between the temperature sensor and the ADC.

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation (up to 45 °C from one chip to another).

The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the temperature sensor measurement, calibration values are stored in system memory for each device by ST during production.

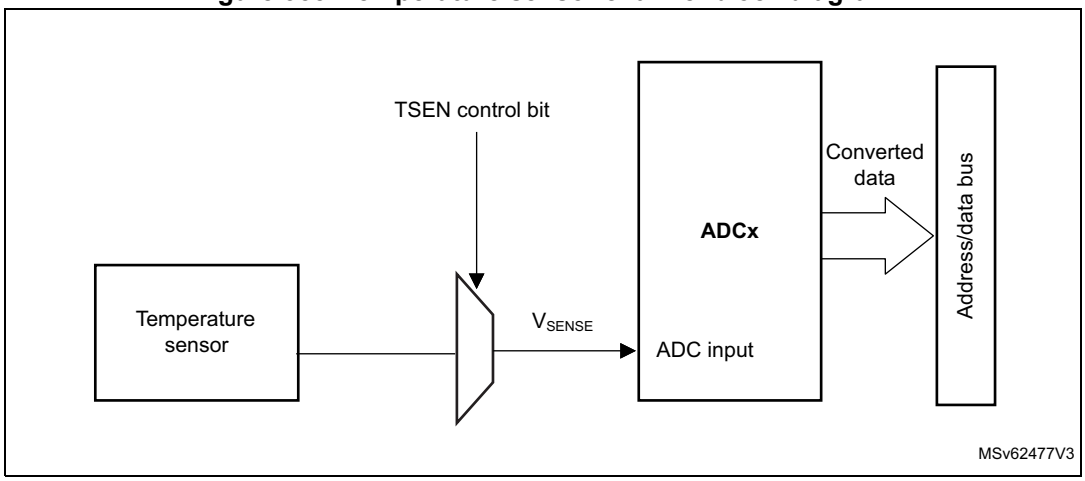
During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference (refer to the datasheet for additional information).

The temperature sensor is internally connected to the ADC input channel, which is used to convert the sensor's output voltage to a digital value. Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal temperature sensor.

When not in use, the sensor can be put in power-down mode.

Figure 355 shows the block diagram of the temperature sensor.

Figure 355. Temperature sensor channel block diagram



Reading the temperature

To use the sensor:

1. Select the ADC input channel that is connected to V_{SENSE}.
2. Program with the appropriate sampling time (refer to the electrical characteristics section of the device datasheet).
3. Set the TSEN bit in the ADCC_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting V_{SENSE} data in the ADC data register.
6. Calculate the actual temperature using the following formula:

$$\text{Temperature (in °C)} = \frac{\text{TS_CAL2_TEMP} - \text{TS_CAL1_TEMP}}{\text{TS_CAL2} - \text{TS_CAL1}} \times (\text{TS_DATA} - \text{TS_CAL1}) + \text{TS_CAL1_TEMP}$$

Where:

- TS_CAL2 is the temperature sensor calibration value acquired at TS_CAL2_TEMP.
- TS_CAL1 is the temperature sensor calibration value acquired at TS_CAL1_TEMP.
- TS_DATA is the actual temperature sensor output value converted by ADC.

  Refer to the device datasheet for more information about TS_CAL1 and TS_CAL2 calibration points.

Caution: The above formula is valid only if the V_{REF+} condition specified in the table *Temperature sensor calibration values* of the datasheet are respected.

Note: The sensor has a startup time after waking from power-down mode before it can output V_{SENSE} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN, and TSEN bits must be set simultaneously.

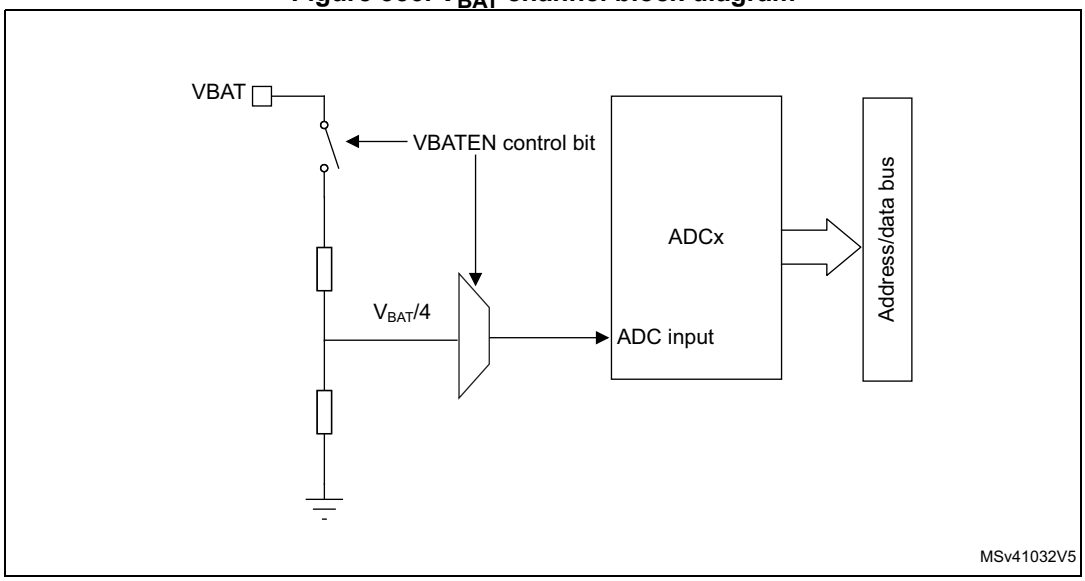
25.4.36 V_{BAT} supply monitoring

The VBATEN bit in the ADCC_CCR register is used to switch to the battery voltage. As the V_{BAT} voltage can be higher than V_{DDA}, to ensure the correct operation of the ADC, the V_{BAT} pin is internally connected to a bridge divider by 4. This bridge is automatically enabled when VBATEN is set, to connect $V_{BAT}/4$ to the ADC input channels (see *Table: ADC interconnection* in *Section 25.4.2: ADC pins and internal signals* for more details). As a consequence, the converted digital value is one fourth of the V_{BAT} voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

Refer to the electrical characteristics of the device datasheet for the sampling time value to be applied when converting the $V_{BAT}/4$ voltage.

Figure 356 shows the block diagram of the V_{BAT} sensing feature.

Figure 356. V_{BAT} channel block diagram



1. The VBATEN bit must be set to enable the conversion of internal channel for $V_{BAT}/4$.

25.4.37 Monitoring the internal voltage reference

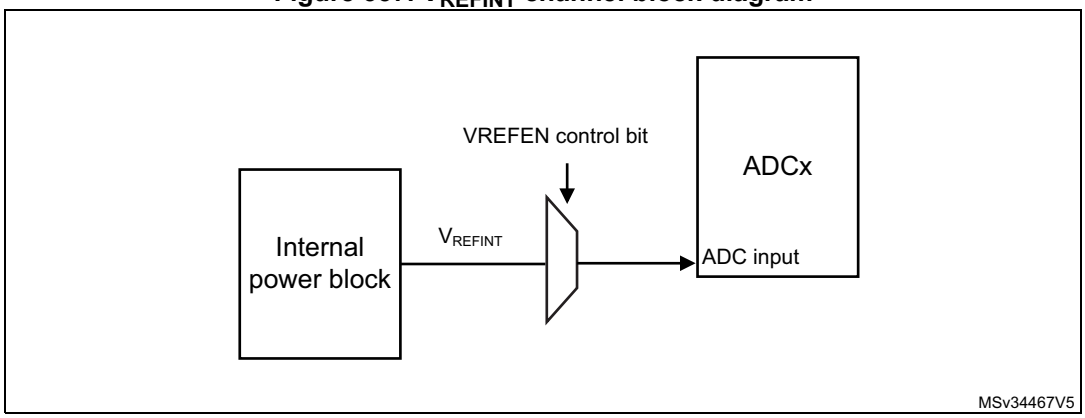
It is possible to monitor the internal voltage reference (V_{REFINT}) to have a reference point for evaluating the ADC V_{REF+} voltage level.

Refer to the *ADC interconnection* table in *Section 25.4.2: ADC pins and internal signals* for details on the ADC input channels to which the internal voltage reference is internally connected.

Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal voltage reference voltage.

Figure 357 shows the block diagram of the V_{REFINT} sensing feature.

Figure 357. V_{REFINT} channel block diagram



1. The VREFEN bit of the ADCC_CCR register must be set to enable the conversion of internal channels (V_{REFINT}).

Calculating the actual V_{REF+} voltage using the internal reference voltage

The power supply voltage applied to the device may be subject to variations or not precisely known. When V_{DDA} is connected to V_{REF+}, it is possible to compute the actual V_{DDA} voltage using the embedded internal reference voltage (V_{REFINT}). V_{REFINT} and its calibration data acquired by the ADC during the manufacturing process at V_{REF+} can be used to evaluate the actual V_{DDA} voltage level.

The following formula gives the actual V_{REF+} voltage supplying the device:

$$V_{REF+} = V_{REF+_Charac} \times VREFINT_CAL / VREFINT_DATA$$

Where:

- V_{REF+_Charac} is the value of V_{REF+} voltage characterized at V_{REFINT} during the manufacturing process. It is specified in the device datasheet.
- VREFINT_CAL is the VREFINT calibration value.
- VREFINT_DATA is the actual VREFINT output value converted by ADC.

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between V_{REF+} and the voltage applied on the converted channel.

For some applications, the V_{REF+} value is unknown and ADC converted values are right-aligned. In this case, it is necessary to convert this ratio into a voltage independent from V_{REF+}:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL_SCALE} \times ADC_DATA$$

By replacing V_{REF+} by the formula provided above, the absolute voltage value is given by the following formula

$$V_{CHANNELx} = \frac{V_{REF+_Charac} \times VREFINT_CAL \times ADC_DATA}{VREFINT_DATA \times FULL_SCALE}$$

For applications where V_{REF+} is known and ADC converted values are right-aligned, the absolute voltage value can be obtained by using the following formula:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL_SCALE} \times ADC_DATA$$

Where:

- VREFINT_CAL is the VREFINT calibration value.
- ADC_DATA is the value measured by the ADC on channel x (right-aligned).
- VREFINT_DATA is the actual VREFINT output value converted by the ADC.
- FULL_SCALE is the maximum digital value of the ADC output. For example with 12-bit resolution, we obtain $2^{12} – 1 = 4095$, or $2^8 – 1 = 255$ with 8-bit resolution.

Note: If ADC measurements are done using an output format other than 12-bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

25.4.38 Monitoring the DAC output voltage

The ADC is connected to a DAC internal channel. To measure the DAC output voltage, enable the DAC internal channel through DAC registers.

25.4.39 Monitoring the supply voltage

The ADCs are connected to the internal supply voltage. To use the ADC to measure this voltage, enable the connection through ADC option register.

25.5 ADC interrupts

For each ADC, an interrupt can be generated:

- After ADC power-up, when the ADC is ready (ADRDY flag)
- On the end of any conversion for regular groups (EOC flag)
- On the end of a sequence of conversion for regular groups (EOS flag)
- On the end of any conversion for injected groups (JEOC flag)
- On the end of a sequence of conversion for injected groups (JEOS flag)
- When an analog watchdog detection occurs (AWD1, AWD2, and AWD3 flags)
- When the end of the sampling phase occurs (EOSMP flag)
- When the data overrun occurs (OVR flag)
- When the ADC internal voltage regulator output is ready (LDORDY flag)
- When the injected sequence context queue overflows (JQOVF flag)

Separate interrupt enable bits are available for flexibility.

Note: Interrupts are generated by the AHB clock domain. Since the ADC clock is adc_ker_ck, interrupt flags are delayed due to the synchronization between adc_ker_ck and AHB clock domains.

Table 254. ADC interrupts

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop, Standby mode
ADC	ADC ready	ADRDY	ADRDYIE	Set by hardware and cleared by software	Yes	No
	End of conversion of a regular group	EOC	EOCIE			
	End of conversion sequence of a regular group	EOS	EOSIE			
	End of conversion of an injected group	JEOC	JEOCIE			
	End of conversion sequence of an injected group	JEOS	JEOSIE			
	Analog watchdog 1 flag is set	AWD1	AWD1IE			
	Analog watchdog 2 flag is set	AWD2	AWD2IE			
	Analog watchdog 3 flag is set	AWD3	AWD3IE			
	End of sampling phase	EOSMP	EOSMPIE			
	Overrun	OVR	OVRIE			
	ADC internal voltage regulator ready	LDORDY	LDORDYIE			
	Injected context queue overflows	JQOVF	JQOVFIE			

25.6 ADC registers (for each ADC)

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

25.6.1 ADC interrupt and status register (ADC_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LDORDY	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
			rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LDORDY**: ADC internal voltage regulator output ready flag

This bit is set by hardware and cleared by software. It indicates that the ADC internal voltage regulator output is ready and that the ADC can be enabled or calibrated.

0: ADC LDO internal voltage regulator disabled

1: ADC LDO internal voltage regulator enabled

Bit 11 Reserved, must be kept at reset value.

Bit 10 **JQOVF**: Injected context queue overflow

This bit is set by hardware when an overflow of the injected queue of context occurs. It is cleared by software writing 1 to it or when ADEN = 0.

Refer to *Section 25.4.23: Queue of context for injected conversions* for more information.

0: No injected context queue overflow occurred (or the flag event was already acknowledged and cleared by software)

1: Injected context queue overflow has occurred

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD3LTR and ADC_AWD3HTR registers. It is cleared by software writing 1 to it or when ADEN = 0.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 3 event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD2LTR and ADC_AWD2HTR registers. It is cleared by software writing 1 to it or when ADEN = 0.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 2 event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD1LTR and ADC_AWD1HTR registers. It is cleared by software writing 1 to it or when ADEN = 0.

0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 1 event occurred

Bit 6 **JEOS:** Injected channel end of sequence flag

This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it or when ADEN = 0.

0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Injected conversions complete

Bit 5 **JEOC:** Injected channel end of conversion flag

This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC_JDRy register. It is cleared by software writing 1 to it, when ADEN = 0, or by reading the corresponding ADC_JDRy register.

0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Injected channel conversion complete

Bit 4 **OVR**: ADC overrun

This bit is set by hardware when an overrun occurs on a regular channel (a new conversion has completed while the EOC flag was already set) or when adc_hclk clock is too slow to manage the data.

It is cleared by software writing 1 to it or when ADEN = 0.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)

1: Overrun has occurred

Bit 3 **EOS**: End of regular sequence flag

This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it or when ADEN = 0.

0: Regular conversions sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Regular conversions sequence complete

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC_DR register. It is cleared by software writing 1 to it, when ADEN = 0, or by reading the ADC_DR register.

0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Regular channel conversion complete

Bit 1 **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase. It is cleared by software writing 1 or when ADEN = 0.

0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

1: End of sampling phase reached

Bit 0 **ADRDY**: ADC ready

This bit is set by hardware after the ADC has been enabled (bit ADEN = 1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it or when ADEN = 0.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

1: ADC is ready to start conversion

25.6.2 ADC interrupt enable register (ADC_IER)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LDORDYIE	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LDORDYIE**: ADC internal voltage regulator interrupt enable

This bit is set and cleared by software to enable/disable the ADC internal voltage regulator interrupt.

0: Internal voltage regulator interrupt disabled

1: Internal voltage regulator interrupt enabled

Note: The software is allowed to write this bit only when ADEN = 0.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **JQOVFIE**: Injected context queue overflow interrupt enable

This bit is set and cleared by software to enable/disable the injected context queue overflow interrupt.

0: Injected context queue overflow interrupt disabled

1: Injected context queue overflow interrupt enabled. An interrupt is generated when the JQOVF bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled

1: Analog watchdog 3 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 2 interrupt disabled

1: Analog watchdog 2 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 1 interrupt.

0: Analog watchdog 1 interrupt disabled

1: Analog watchdog 1 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.

0: JEOS interrupt disabled

1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 5 **JEOCIE**: End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.

0: JEOC interrupt disabled.

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 4 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 3 **EOSIE**: End of regular sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of regular sequence of conversions interrupt.

0: EOS interrupt disabled

1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 2 **EOCIE**: End of regular conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of a regular conversion interrupt.

0: EOC interrupt disabled.

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable for regular conversions

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.

0: EOSMP interrupt disabled.

1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 0 **ADRDYIE**: ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

0: ADRDY interrupt disabled

1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

25.6.3 ADC control register (ADC_CR)

Address offset: 0x008

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	ADCALDIF	DEEPPWD	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
										rs	rs	rs	rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of the ADC. Program first the bit ADCALDIF to determine if this calibration applies for single-ended or differential inputs mode.

It is cleared by hardware after calibration is complete.

0: Calibration complete

1: Write 1 to calibrate the ADC. Read at 1 means that a calibration in progress.

Note: The software is allowed to launch a calibration by setting ADCAL only when ADEN = 0.

The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing)

Bit 30 **ADCALDIF**: Differential mode for calibration

This bit is set and cleared by software to configure the single-ended or differential input mode for the calibration.

0: Single-ended input calibration mode when ADCAL = 1

1: Differential input calibration mode when ADCAL = 1

Note: The software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bit 29 **DEEPPWD**: Deep-power-down enable

This bit is set and cleared by software to put the ADC in Deep-power-down mode.

0: ADC not in Deep-power down

1: ADC in Deep-power-down (default reset state)

Note: The software is allowed to write this bit only when the ADC is disabled (ADCAL = 0 and ADEN =0).

Bit 28 **ADVREGEN**: ADC internal voltage regulator enable

This bits is set by software to enable the ADC internal voltage regulator.

Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

0: ADC internal voltage regulator disabled

1: ADC internal voltage regulator enabled.

For more details about the ADC voltage regulator enable and disable sequences, refer to *Section 25.4.6: ADC Deep-power-down mode (DEEPPWD)*.

The software can program this bit field only when the ADC is disabled (ADCAL = 0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 27:6 Reserved, must be kept at reset value.

Bit 5 **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP command).

It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be reconfigured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

When a software trigger is used, JADSTART bit is cleared by hardware, but JADSTP bit must be programmed to reconfigure the ADC.

0: No ADC stop injected conversion command ongoing

1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: In autoinjection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).

Bit 4 **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP command).

It is cleared by hardware when the conversion is effectively discarded. and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

When a software trigger is used, ADSTART bit is cleared by hardware. However it is necessary to write ADSTP bit to reconfigure the ADC.

0: No ADC stop regular conversion command ongoing

1: Write 1 to stop ongoing regular conversions. Read 1 means that an ADSTP command is in progress.

Note: In autoinjection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use ADSTP).

Bit 3 **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN[1:0], a conversion starts immediately (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

- In single conversion mode when software trigger is selected (JEXTSEL = 0x0 and JDISEN = 1), this bit is cleared by hardware immediately after injected conversion starts.
- In all cases, after the execution of the JADSTP command, this bit and JADSTP are cleared by hardware simultaneously.

0: No ADC injected conversion is ongoing.

1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

Note: The software is allowed to set JADSTART only when ADEN = 1, ADDIS = 0 and JAUTO = 0 (ADC is enabled and there is no pending request to disable the ADC).

Note: In autoinjection mode (JAUTO = 1), regular and autoinjected conversions are started by setting ADSTART bit (JADSTART must be kept cleared).

Bit 2 **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN[1:0], a conversion starts immediately (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

- In single conversion mode when software trigger is selected (EXTSEL = 0x0 and DISEN = 1), this bit is cleared by hardware immediately after regular conversion starts.
- In all cases, after the execution of the ADSTP command, this bit and ADSTP are cleared by hardware simultaneously.

0: No ADC regular conversion is ongoing.

1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

Note: The software is allowed to set ADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).

Note: In autoinjection mode (JAUTO = 1), regular and autoinjected conversions are started by setting ADSTART bit (JADSTART must be kept cleared)

Bit 1 **ADDIS**: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and place it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: no ADDIS command ongoing

1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

Note: The software is allowed to set ADDIS only when ADEN = 1, ADDIS = 0 and both ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 0 **ADEN**: ADC enable control

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the ADRDY flag has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

Note: The software is allowed to set ADEN only when all the bits of ADC_CR registers are cleared (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0), except for ADVREGEN bit which must be set. In addition, the software must wait for the startup time of the voltage regulator.

25.6.4 ADC configuration register (ADC_CFGR1)

Address offset: 0x00C

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM[2:0]			DISCEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL[4:0]					Res.	RES[1:0]		DMNGT[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bit 31 **JQDIS**: Injected queue disable

This bit is set and cleared by software to disable the injected queue mechanism:

0: Injected queue enabled

1: Injected queue disabled

Note: The software is allowed to write this bit only when ADEN = 0, ADSTART = 0, and JADSTART = 0.

Note: A set or reset of JQDIS bit causes the injected queue to be flushed and the ADC_JSQR register is cleared.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog 1 channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input channel 0 monitored by AWD1

00001: ADC analog input channel 1 monitored by AWD1

.....

10011: ADC analog input channel 19 monitored by AWD1

Others: reserved, must not be used

Note: Some channels are not connected physically. Keep the corresponding AWD1CH[4:0] setting to the reset value.

Note: The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers.

The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **JAUTO:** Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled

1: Automatic injected group conversion enabled

Note: The software is allowed to write this bit only when ADEN = 0.

Bit 24 **JAWD1EN**: Analog watchdog 1 enable on injected channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on injected channels

1: Analog watchdog 1 enabled on injected channels

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on regular channels

1: Analog watchdog 1 enabled on regular channels

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 22 **AWD1SGL**: Enable the watchdog 1 on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWD1CH[4:0] bits or on all the channels

0: Analog watchdog 1 enabled on all channels

1: Analog watchdog 1 enabled on a single channel

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 21 **JQM**: ADC_JSQR queue mode

This bit is set and cleared by software.

It defines how an empty Queue is managed.

0: ADC_JSQR mode 0: The Queue is never empty and maintains the last written configuration to ADC_JSQR.

1: ADC_JSQR mode 1: The Queue can be empty and when this occurs, the software and hardware triggers of the injected sequence are both internally disabled just after the completion of the last valid injected sequence.

Refer to *Section 25.4.23: Queue of context for injected conversions* for more information.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 20 **JDISCEN:** Discontinuous mode on injected channels

This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.

0: Discontinuous mode on injected channels disabled

1: Discontinuous mode on injected channels enabled

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Note: It is not possible to use autoinjection mode and discontinuous mode simultaneously: the DISCEN and JDISCEN bits must be kept cleared by software when JAUTO is set.

Note: When dual mode is enabled (DUAL bits of ADCC_CCR register are not equal to zero), the JDISCEN bit of the slave ADC is not valid and controlled by the JDISCEN bit of the master ADC.

Bits 19:17 **DISCNUM[2:0]:** Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel

001: 2 channels

...

111: 8 channels

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: When dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the DISCNUM[2:0] bits of the slave ADC are not valid and controlled by the DISCNUM[2:0] bits of the master ADC.

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable discontinuous mode for regular channels.

0: Discontinuous mode for regular channels disabled

1: Discontinuous mode for regular channels enabled

Note: It is not possible to have both discontinuous mode and continuous mode enabled: DISCEN and CONT cannot both be set (DISCEN takes priority).

Note: It is not possible to use autoinjection mode and discontinuous mode simultaneously: the DISCEN and JDISCEN bits must be kept cleared by software when JAUTO is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: When regular dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the DISCEN bit of the slave ADC is not valid and controlled by the DISCEN bit of the master ADC.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the autodelayed conversion mode.

0: Autodelayed conversion mode off

1: Autodelayed conversion mode on

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: When dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the AUTDLY bit of the slave ADC is not valid and controlled by the AUTDLY bit of the master ADC.

Bit 13 **CONT**: Single/continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

Note: It is not possible to have both discontinuous mode and continuous mode enabled: DISCEN and CONT cannot both be set (DISCEN takes priority).

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: When regular dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the CONT bit of the slave ADC is not valid and controlled by the CONT bit of the master ADC.

Bit 12 **OVRMOD**: Overrun mode

This bit is set and cleared by software and configure the way data overrun is managed.

0: ADC_DR register is preserved with the old data when an overrun is detected.

1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 9:5 **EXTSEL[4:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

00000: adc_ext_trg0
00001: adc_ext_trg1
00010: adc_ext_trg2
00011: adc_ext_trg3
00100: adc_ext_trg4
00101: adc_ext_trg5
00110: adc_ext_trg6
00111: adc_ext_trg7
...
11111: adc_ext_trg31

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

00: 12-bit
01: 10-bit
10: 8-bit
11: 6-bit

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 1:0 **DMNGT[1:0]**: Data management configuration

These bits are set and cleared by software to select how the ADC interface output data are managed.

00: Regular conversion data stored in DR only
01: DMA one-shot mode selected
10: ADF mode selected
11: DMA circular mode selected

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).

25.6.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LSHIFT[3:0]				Res.	Res.	OVSR[9:0]									
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMPTRIG	SWTRIG	BULB	Res.	Res.	ROVSM	TROVS	OVSS[3:0]				Res.	PDIFFREF	PDIFF	Res.	ROVSE
rw	rw	rw			rw	rw	rw	rw	rw	rw		rw	rw		rw

Bits 31:28 **LSHIFT[3:0]**: Left shift factor

This bitfield is set and cleared by software to define the left shifting applied to the final result with or without oversampling.

0000: No left shift
0001: 1-bit left shift
0010: 2-bit left shift
0011: 3-bit left shift
0100: 4-bit left shift
0101: 5-bit left shift
0110: 6-bit left shift
0111: 7-bit left shift
1000: 8-bit left shift
1001: 9-bit left shift
1010: 10-bit left shift
1011: 11-bit left shift
1100: 12-bit left shift
1101: 13-bit left shift
1110: 14-bit left shift
1111: 15-bit left shift

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:16 **OVSR[9:0]**: Oversampling ratio

This bitfield is set and cleared by software to define the oversampling ratio.

0: 1x (no oversampling)
1: 2x
2: 3x
...
1023: 1024x

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 15 **SMPTRIG**: Sampling time control trigger mode

This bit is set and cleared by software to enable the sampling time control trigger mode.

0: Sampling time control trigger mode disabled
1: Sampling time control trigger mode enabled

If EXTEN[1:0] bits are set to 01, the sampling time starts on the trigger rising edge, and the conversion starts on the trigger falling edge.

The SMPTRIG bit must not be set when the BULB bit is set.

When EXTEN[1:0] bits are set to 00, set SWTRIG to start the sampling.

Note: The software is allowed to write this bit only when ADEN = 0. When the discontinuous mode is used, only DISCNUM[2:0] = 000 configuration is compatible with sampling time control trigger mode*.*

Bit 14 **SWTRIG**: Software trigger bit for sampling time control trigger mode

This bit is set and cleared by software to trigger the conversion in sampling time control trigger mode.

0: Software trigger starts the conversion for sampling time control trigger mode
1: Software trigger starts the sampling for *sampling time control trigger mode*

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing), SMPTRIG = 1, and EXTEN[0:1] = 00.

Bit 13 **BULB**: Bulb sampling mode

This bit is set and cleared by software to enable the bulb sampling mode.

0: Bulb sampling mode disabled

1: Bulb sampling mode enabled. The sampling period starts just after the previous end of conversion.

BULB bit must not be set when the SWTRIG bit is set.

The very first ADC conversion is performed with the sampling time specified in SMPx bits.

Note: The software is allowed to write this bit only when ADEN = 0. When the discontinuous mode is used, only DISCNUM[2:0] = 000 configuration is compatible with bulb mode.

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **ROVSM**: Regular oversampling mode

This bit is set and cleared by software to select the regular oversampling mode.

0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injected sequence)

1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injected sequence start)

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing). It is recommended to clear both JOVSE and GCOMP when ROVSM = 0.

Bit 9 **TROVS**: Triggered regular oversampling

This bit is set and cleared by software to enable triggered oversampling

0: All oversampled conversions for a channel are done consecutively following a trigger

1: Each oversampled conversion for a channel needs a new trigger

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 8:5 **OVSS[3:0]**: Oversampling shift

This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.

0000: No shift

0001: 1-bit shift

0010: 2-bit shift

0011: 3-bit shift

0100: 4-bit shift

0101: 5-bit shift

0110: 6-bit shift

0111: 7-bit shift

1000: 8-bit shift

1001: 9-bit shift

1010: 10-bit shift

Other: reserved, must not be used

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 4 Reserved, must be kept at reset value.

Bit 3 **PDIFFREF**: Pseudodifferential mode reference voltage selection
This bit is set and cleared by software to select the reference voltage used in pseudodifferential mode.
0: V_{REF-}
1: $V_{REF+}/2$
Note: The software is allowed to write this bit only when ADEN = 0.

Bit 2 **PDIFF**: Pseudodifferential mode enable
This bit is set and cleared by software to enable/disable the pseudodifferential mode.
0: Standard differential mode selected
1: Pseudodifferential mode selected
Note: The software is allowed to write this bit only when ADEN = 0.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ROVSE**: Regular oversampling enable
This bit is set and cleared by software to enable regular oversampling.
0: Regular oversampling disabled
1: Regular oversampling enabled
Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing)

25.6.6 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5[0]	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 9 to 0)
These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.
000: 2.5 ADC clock cycles
001: 4.5 ADC clock cycles
010: 7.5 ADC clock cycles
011: 12.5 ADC clock cycles
100: 24.5 ADC clock cycles
101: 47.5 ADC clock cycles
110: 256.5 ADC clock cycles
111: 848.5 ADC clock cycles
Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
Note: Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

25.6.7 ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SMP19[2:0]			SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15[0]	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 19 to 10)

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

000: 2.5 ADC clock cycles
001: 4.5 ADC clock cycles
010: 7.5 ADC clock cycles
011: 12.5 ADC clock cycles
100: 24.5 ADC clock cycles
101: 47.5 ADC clock cycles
110: 256.5 ADC clock cycles
111: 848.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

25.6.8 ADC channel preselection register (ADC_PCSEL)

Address offset: 0x01C

Reset value: 0x0000 0000

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

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **PCSEL[19:0]**: Channel i ($V_{INP[i]}$) preselection

These bits are written by software to preselect the input channel I/O to be converted.

0: Input channel i ($V_{INP[i]}$) is not preselected for conversion, the result of the ADC conversion for this channel is wrong

1: Input channel i ($V_{INP[i]}$) is preselected for conversion

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Configuring the PCSEL bit is not necessary for the internal channels (such as VREFINT).

25.6.9 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ2[3:0]				Res.	SQ1[4:0]					Res.	Res.	LEN[3:0]			
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]:** 4th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 4th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ3[4:0]:** 3rd conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 3rd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]:** 2nd conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 2nd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]:** 1st conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 1st in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **LEN[3:0]:** Regular channel sequence length
These bits are written by software to define the total number of conversions in the regular channel conversion sequence.
0000: 1 conversion
0001: 2 conversions
...
1111: 16 conversions
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

25.6.10 ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ7[3:0]				Res.	SQ6[4:0]					Res.	SQ5[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]:** 9th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 9th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]:** 8th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 8th in the regular conversion sequence
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]:** 7th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 7th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]:** 6th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 6th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]:** 5th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 5th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

25.6.11 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ12[3:0]				Res.	SQ11[4:0]					Res.	SQ10[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ14[4:0]:** 14th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 14th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ13[4:0]:** 13th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 13th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ12[4:0]:** 12th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 12th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ11[4:0]:** 11th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 11th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ10[4:0]:** 10th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 10th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

25.6.12 ADC regular sequence register 4 (ADC_SQR4)

Address offset: 0x03C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
					rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:6 **SQ16[4:0]:** 16th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 16th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ15[4:0]:** 15th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 15th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

25.6.13 ADC regular data register (ADC_DR)

Address offset: 0x040

Reset value: 0x0000 0000

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

Bits 31:0 **RDATA[31:0]**: Regular data converted

These bits are read-only. They contain the conversion result from the last converted regular channel. The data are left- or right-aligned as described in *Section 25.4.28: Data management*.

25.6.14 ADC injected sequence register (ADC_JSQR)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:1]			
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ2[0]	Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JLEN[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 **JSQ4[4:0]:** 4th conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 4th in the injected conversion sequence.

Note: The software is allowed to write these bits only when JQDIS = 0 or when JADSTART = 0 and JQDIS = 1 (which ensures that no injected conversion is ongoing).

Bit 26 Reserved, must be kept at reset value.

Bits 25:21 **JSQ3[4:0]:** 3rd conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 3rd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JQDIS = 0 or when JADSTART = 0 and JQDIS = 1 (which ensures that no injected conversion is ongoing).

Bit 20 Reserved, must be kept at reset value.

Bits 19:15 **JSQ2[4:0]:** 2nd conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 2nd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JQDIS = 0 or when JADSTART = 0 and JQDIS = 1 (which ensures that no injected conversion is ongoing).

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **JSQ1[4:0]:** 1st conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 1st in the injected conversion sequence.

Note: The software is allowed to write these bits only when JQDIS = 0 or when JADSTART = 0 and JQDIS = 1 (which ensures that no injected conversion is ongoing).

Bits 8:7 **JEXTEN[1:0]**: External trigger enable and polarity selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when JQDIS = 0 or when JADSTART = 0 and JQDIS = 1 (which ensures that no injected conversion is ongoing).

Bits 6:2 **JEXTSEL[4:0]**: External trigger selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

00000: adc_jext_trg0

00001: adc_jext_trg1

00010: adc_jext_trg2

00011: adc_jext_trg3

00100: adc_jext_trg4

00101: adc_jext_trg5

00110: adc_jext_trg6

00111: adc_jext_trg7

...

11111: adc_jext_trg31

Note: The software is allowed to write these bits only when JQDIS = 0 or when JADSTART = 0 and JQDIS = 1 (which ensures that no injected conversion is ongoing).

Bits 1:0 **JLEN[1:0]:** Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

Note: The software is allowed to write these bits only when JQDIS = 0 or when JADSTART = 0 and JQDIS = 1 (which ensures that no injected conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

25.6.15 ADC offset y configuration register (ADC_OFCFGRy)

Address offset: 0x050 + 0x004 * (y -1), (y = 1 to 4)

Reset value: 0x0000 0000

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

Bits 31:27 **OFFSET_CH[4:0]:** Channel selection for the data offset y

These bits are written by software to define the channel to which the offset programmed in bits OFFSET[21:0] applies.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the data offset y.

Bit 26 **SSAT**: Signed saturation enable

This bit is set and cleared by software to enable the signed saturation feature.(see *Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS, JLSHIFT[a], USAT, SSAT, POSOFF)*).

0: Offset is subtracted maintaining the data integrity and extending converted data size (13-bit signed format)

1: Offset is subtracted and result is saturated to maintain the converted data size

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **USAT**: Unsigned saturation enable

This bit is set and cleared by software to enable the unsigned saturation feature.

0: Offset is subtracted maintaining the data integrity

1: Offset is subtracted and result is saturated to maintain the converted data size

Note: The software is allowed to write these bits only when ADSTART = and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 24 **POSOFF**: Positive offset enable

This bit is set and cleared by software to enable the positive offset

0: Negative offset

1: Positive offset

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

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

25.6.16 ADC offset y register (ADC_OFRy)

Address offset: 0x060 + 0x004 * (y -1), (y= 1 to 4)

Reset value: 0x0000 0000

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

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

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

Bits 21:0 **OFFSET[21:0]:** Data offset y for the channel programmed in OFFSETy_CH[4:0] bits

These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset y must be programmed in the OFFSETy_CH[4:0] bits. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

25.6.17 ADC gain compensation register (ADC_GCOMP)

Address offset: 0x070

Reset value: 0x0000 1000

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

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

Bit 31 **GCOMP:** Gain compensation mode

This bit is set and cleared by software to enable the Gain compensation mode.

0: Regular ADC operation mode

1: Gain compensation enabled and applied on all channels.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensure that no conversion is ongoing)

Bits 30:14 Reserved, must be kept at reset value.

Bits 13:0 **GCOMPCOEFF[13:0]:** Gain compensation coefficient

These bits are set and cleared by software to program the gain compensation coefficient.

00 1000 0000 0000: gain factor of 0.5

...

01 0000 0000 0000: gain factor of 1

10 0000 0000 0000: gain factor of 2

11 0000 0000 0000: gain factor of 3

...

The coefficient is divided by 4096 to get the gain factor ranging from 0 to 3.9999756

Note: This gain compensation is only applied when GCOMP bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensure that no conversion is ongoing).

25.6.18 ADC analog watchdog y source channel register (ADC_AWDySR)

Address offset: 0x070 + 0x004 * (y -1), (y = 1 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDS_CH[19:16]			
												rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWDS_CH[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWDS_CH[19:0]:** Analog watchdog source for channel *i* (V_{INP}[i])

These bits are set by hardware when channel *i* has generated the flag. They are cleared by software by writing 1 to it, or by setting the ADDIS bit. Each bit corresponds to the input channel that created the analog watchdog flag.

25.6.19 ADC injected channel y data register (ADC_JDRy)

Address offset: 0x080 + 0x004 * (y - 1), (y = 1 to 4)

Reset value: 0x0000 0000

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

Bits 31:0 **JDATA[31:0]:** Injected data

These bits are read-only. They contain the conversion result from injected channel y. The data are left- or right-aligned as described in *Section 25.4.28: Data management*.

25.6.20 ADC configuration register 3 (ADC_CFGR3)

Address offset: 0x090

Reset value: 0x0000 0000

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

Bits 31:28 **JLSHIFT[3:0]**: Left shift factor

This bitfield is set and cleared by software to define the left shifting applied to the final result with or without oversampling.

0000: No left shift
0001: 1-bit left shift
0010: 2-bit left shift
0011: 3-bit left shift
0100: 4-bit left shift
0101: 5-bit left shift
0110: 6-bit left shift
0111: 7-bit left shift
1000: 8-bit left shift
1001: 9-bit left shift
1010: 10-bit left shift
1011: 11-bit left shift
1100: 12-bit left shift
1101: 13-bit left shift
1110: 14-bit left shift
1111: 15-bit left shift

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:16 **JOVSR[9:0]**: Injected channel oversampling ratio

This bitfield is set and cleared by software to define the injected channel oversampling ratio.

00 0000 0000: 1x (no oversampling)

00 0000 0001-11 1111 1111: {val+1}x

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **TJOVS**: Triggered injected channel oversampling

This bit is set and cleared by software to enable triggered injected channel oversampling.

0: All oversampled conversions for a channel are done consecutively following a trigger
1: Each oversampled conversion for a channel needs a new trigger

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 8:5 **JOVSS[3:0]**: Injected channel oversampling shift

This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.

0000: No shift
0001: Shift 1-bit
0010: Shift 2-bits
0011: Shift 3-bits
0100: Shift 4-bits
0101: Shift 5-bits
0110: Shift 6-bits
0111: Shift 7-bits
1000: Shift 8-bits
Others: reserved

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **JOVSIVE**: Injected oversampling intermediate value enable

This bit is set and cleared by software to enable the access to the injected oversampling intermediate value on JDATA register.

0: Injected oversampling intermediate data not available on JDATA
1: Injected oversampling intermediate data available on JDATA

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 1 **JOVSE**: Injected oversampling enable

This bit is set and cleared by software to enable injected oversampling.

0: Injected oversampling disabled
1: Injected oversampling enabled

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 0 **JOVSPAR**: Parallel injected oversampling enable

This bit is set and cleared by software to enable parallel injected oversampling.

0: Parallel Injected oversampling disabled
1: Parallel Injected oversampling enabled

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no conversion is ongoing), it is necessary to set JOVSE bit when JOVSPAR bit set.

25.6.21 ADC injected oversampling counter register 1 (ADC_JOVSCR1)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	JOVSC2[9:0]									
						r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	JOVSC1[9:0]									
						r	r	r	r	r	r	r	r	r	r

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

Bits 25:16 **JOVSC2[9:0]:** Oversampling counter number for injected channel 2
This bitfield is set by hardware and cleared by hardware when JADSTART or ADDIS is set.
00 0000 0000: No oversampling or oversampling conversion finished
00 0000 0001 - 11 1111 1111: Number of remaining conversions to complete the oversampling

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **JOVSC1[9:0]:** Oversampling counter number for injected channel 1
This bitfield is set by hardware and cleared by hardware when JADSTART or ADDIS is set.
00 0000 0000: No oversampling or oversampling conversion finished
00 0000 0001 - 11 1111 1111: Number of remaining conversions to complete the oversampling

25.6.22 ADC injected oversampling counter register 2 (ADC_JOVSCR2)

Address offset: 0x098

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	JOVSC4[9:0]									
						r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	JOVSC3[9:0]									
						r	r	r	r	r	r	r	r	r	r

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

Bits 25:16 **JOVSC4[9:0]:** Oversampling counter number for injected channel 4
This bitfield is set by hardware and cleared by hardware when JADSTART or ADDIS is set.
00 0000 0000: Number of remaining conversions to complete the oversampling
00 0000 0001 - 11 1111 1111: Number of remaining conversions to complete the oversampling

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **JOVSC3[9:0]:** Oversampling counter number for injected channel 3
This bitfield is set by hardware and cleared by hardware when JADSTART or ADDIS is set.
00 0000 0000: No oversampling or oversampling conversion finished
00 0000 0001 - 11 1111 1111: Number of remaining conversions to complete the oversampling

25.6.23 ADC analog watchdog 2 configuration register (ADC_AWD2CR)

Address offset: 0x0A0

Reset value: 0x0000 0000

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

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWDCH[19:0]**: Analog watchdog 2 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.

AWDCH[i] = 0: ADC analog input channel i is not monitored by AWD2

AWDCH[i] = 1: ADC analog input channel i is monitored by AWD2

When AWDCH[19:0] = 000..0, the analog Watchdog 2 is disabled

Note: The channels selected by AWDCH must be also selected in the SQi or JSQi bits.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the analog watchdog.

25.6.24 ADC analog watchdog 3 configuration register (ADC_AWD3CR)

Address offset: 0x0A4

Reset value: 0x0000 0000

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

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWDCH[19:0]**: Analog watchdog 3 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.

AWDCH[i] = 0: ADC analog input channel i is not monitored by AWD3

AWDCH[i] = 1: ADC analog input channel i is monitored by AWD3

When AWDCH[19:0] = 000..0, the analog Watchdog 3 is disabled.

Note: The channels selected by AWDCH must be also selected in the SQi or JSQi bits.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the analog watchdog.

25.6.25 ADC analog watchdog 1 lower threshold register (ADC_AWD1LTR)

Address offset: 0x0A8

Reset value: 0x0000 0000

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

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

Bits 22:0 **LTR[22:0]**: Analog watchdog 1 lower threshold

These bits are set and cleared by software to define the lower threshold for analog watchdog 1.
Refer to *Section 25.4.31: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)*.

25.6.26 ADC analog watchdog 1 higher threshold register (ADC_AWD1HTR)

Address offset: 0x0AC

Reset value: 0x003F FFFF

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

Bits 31:29 **AWDFILT[2:0]**: Analog watchdog filtering parameter

These bits are set and cleared by software.
000: No filtering, one detection generates an AWD1 flag or an interrupt
001: two consecutive detections generate an AWD1 flag or an interrupt
...
111: Eight consecutive detections generate an AWD1 flag or an interrupt

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 1 higher threshold

These bits are set and cleared by software to define the higher threshold for analog watchdog 1.
Refer to *Analog windows watchdog* section.

25.6.27 ADC analog watchdog 2 lower threshold register (ADC_AWD2LTR)

Address offset: 0x0B0

Reset value: 0x0000 0000

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

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

Bits 22:0 **LTR[22:0]**: Analog watchdog 2 lower threshold

These bits are set and cleared by software to define the lower threshold for analog watchdog 2. Refer to *Analog windows watchdog* section.

25.6.28 ADC analog watchdog 2 higher threshold register (ADC_AWD2HTR)

Address offset: 0x0B4

Reset value: 0x003F FFFF

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

Bits 31:29 **AWDFILT[2:0]**: Analog watchdog filtering parameter

These bits are set and cleared by software.

000: No filtering, one detection generates an AWD2 flag or an interrupt

001: two consecutive detections generate an AWD2 flag or an interrupt

...

111: Eight consecutive detections generate an AWD2 flag or an interrupt

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 2 higher threshold

These bits are set and cleared by software to define the higher threshold for analog watchdog 2. Refer to *Analog windows watchdog* section.

25.6.29 ADC analog watchdog 3 lower threshold register (ADC_AWD3LTR)

Address offset: 0x0B8

Reset value: 0x0000 0000

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

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

Bits 22:0 **LTR[22:0]**: Analog watchdog 3 lower threshold

These bits are set and cleared by software to define the lower threshold for analog watchdog 3. Refer to *Analog windows watchdog* section.

25.6.30 ADC analog watchdog 3 higher threshold register (ADC_AWD3HTR)

Address offset: 0x0BC

Reset value: 0x003F FFFF

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

Bits 31:29 **AWDFILT[2:0]**: Analog watchdog filtering parameter

These bits are set and cleared by software.

000: No filtering, one detection generates an AWD3 flag or an interrupt

001: two consecutive detections generate an AWD3 flag or an interrupt

...

111: Eight consecutive detections generate an AWD3 flag or an interrupt

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 3 higher threshold

These bits are set and cleared by software to define the higher threshold for analog watchdog 3. Refer to *Analog windows watchdog* section.

25.6.31 ADC differential mode selection register (ADC_DIFSEL)

Address offset: 0x0C0

Reset value: 0x0000 0000

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

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **DIFSEL[19:0]:** Differential mode for channels 19 to 0.

These bits are set and cleared by software. They allow to select if a channel is configured as single-ended or differential or pseudodifferential mode.

DIFSEL[i] = 0: ADC analog input channel is configured in single-ended mode

DIFSEL[i] = 1: ADC analog input channel i is configured in differential or pseudodifferential mode

Note: The DIFSEL bits corresponding to channels that are either connected to a single-ended I/O port or to an internal channel must be kept their reset value (single-ended input mode).

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

25.6.32 ADC calibration factors (ADC_CALFACT)

Address offset: 0x0C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_D[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

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

Bits 22:16 **CALFACT_D[6:0]**: Calibration factors in differential mode

These bits are written by hardware or by software.

Once a differential input calibration is complete, they are updated by hardware with the calibration factors.

The software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored in the analog ADC, it is applied once a new differential conversion is launched.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (ADC is enabled, no calibration is ongoing and no conversion is ongoing).

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

Bits 6:0 **CALFACT_S[6:0]**: Calibration factors In single-ended mode

These bits are written by hardware or by software.

Once a single-ended input calibration is complete, they are updated by hardware with the calibration factors.

The software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored in the analog ADC, it is applied once a new single-ended conversion is launched.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (ADC is enabled, no calibration is ongoing and no conversion is ongoing).

This bit can only be cleared by a power-on reset.

25.6.33 ADC option register (ADC_OR)

Address offset: 0x0D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OP7	OP6	OP5	OP4	OP3	OP2	OP1	OP0
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 7 **OP7**: Option bit 7

This bit is set and cleared by software to enable/disable the internal reference buffer dynamic biasing.

0: Enable VCM dynamic biasing

1: Disable VCM dynamic biasing

Note: The software is allowed to write this bit only when the ADC is disabled.

Bit 6 **OP6:** Option bit 6

This bit is set and cleared by software to select the redundancy bit.

0: Redundancy on bit 5

1:Redundancy on bit 3

Note: The software is allowed to write this bit only when the ADC is disabled.

Bit 5 **OP5:** Option bit 5

This bit is set and cleared by software to enabled/disable the redundancy.

0: Redundancy disabled

1:Redundancy enabled

Note: The software is allowed to write this bit only when the ADC is disabled.

Bit 4 **OP4:** Option bit 4

This bit is set and cleared by software to select the optional trigger source

Condition: ADC1 and ADC2

0: Reserved

1: Reserved

Condition: ADC3 and ADC4

0: adc_jext_trg9 = tim8_oc4

1: adc_jext_trg9 = tim4_trgo

Note: If one of the OP4 bit is set in ADC3 or ADC4, tim4_trgo is selected for both ADC3 and ADC4.

Note: The software is allowed to write this bit only when the JADSTART = 0.

Bit 3 **OP3:** Option bit 3

This bit is set and cleared by software to select the optional trigger source.
Condition: ADC1 and ADC2
0: adc_jext_trg9 = tim3_trgo
1: adc_jext_trg9 = tim4_trgo

Note: If one of the OP3 bit is set in ADC1 or ADC2, tim4_trgo is selected for both ADC1 and ADC2.

Condition: ADC3 and ADC4
0: adc_jext_trg8 = tim8_oc2
1: adc_jext_trg8 = tim4_oc3

Note: If one of the OP3 bit is set in ADC3 or ADC4, tim4_oc3 is selected for both ADC3 and ADC4.

Note: The software is allowed to write this bit only when the JADSTART = 0.

Bit 2 **OP2:** Option bit 2

This bit is set and cleared by software to select the optional trigger source.
Condition: ADC1 and ADC2
0: adc_ext_trg9 = tim3_trgo
1: adc_ext_trg9 = tim4_trgo

Note: If one of the OP2 bit is set in ADC1 or ADC2, tim4_trgo is selected for both ADC1 and ADC2.

Condition: ADC3 and ADC4
0: adc_ext_trg9 = tim3_trgo
1: adc_ext_trg9 = tim4_trgo

Note: If one of the OP2 bit is set in ADC3 or ADC4, tim4_trgo is selected for both ADC3 and ADC4.

Note: The software is allowed to write this bit only when the ADSTART = 0 and no regular conversion is ongoing.

Bit 1 **OP1:** Option bit 1

This bit is set and cleared by software to select the optional trigger source
Condition: ADC1 and ADC2
0: adc_ext_trg8 = tim3_oc4
1: adc_ext_trg8 = tim4_oc4

Note: If one of the OP1 bit is set in ADC1 or ADC2, tim4_oc4 is selected for both ADC1 and ADC2.

Condition: ADC3 and ADC4
0: adc_ext_trg8 = tim3_oc2
1: adc_ext_trg8 = tim4_oc1

Note: If one of the OP1 bit is set in ADC3 or ADC4, tim4_oc1 is selected for both ADC3 and ADC4.

Note: The software is allowed to write this bit only when the ADSTART = 0 and no regular conversion is ongoing.

Bit 0 **OP0:** Option bit 0

This bit is set and cleared by software to enable/disable the V_{DDCORE} channel.
Condition: ADC1 and ADC2 and ADC4
0: Reserved
1: Reserved
Condition: ADC3
0: V_{DDCORE} disabled
1: V_{DDCORE} enabled

25.7 ADC common registers

These registers define the control and status registers common to master and slave ADCs.

25.7.1 ADC common status register (ADCC_CSR)

Address offset: 0x000

Reset value: 0x0000 0000

This register provides an image of the flags of the different ADCs. Nevertheless it is read-only and does not allow to clear the different flags. Instead each flag must be cleared by writing 0 to it in the corresponding ADC_ISR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	LDORDY_SLV	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV
			r		r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LDORDY_MST	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
			r		r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **LDORDY_SLV:** ADC internal voltage regulator flag of the slave ADC
This bit is a copy of the LDORDY bit in the corresponding ADC_ISR register.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **JQOVF_SLV:** Injected context queue overflow flag of the slave ADC
This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 25 **AWD3_SLV:** Analog watchdog 3 flag of the slave ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 24 **AWD2_SLV:** Analog watchdog 2 flag of the slave ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 23 **AWD1_SLV:** Analog watchdog 1 flag of the slave ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 22 **JEOS_SLV:** End of injected sequence flag of the slave ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 21 **JEOC_SLV:** End of injected conversion flag of the slave ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 20 **OVR_SLV:** Overrun flag of the slave ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 19 **EOS_SLV:** End of regular sequence flag of the slave ADC. This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 18 **EOC_SLV:** End of regular conversion of the slave ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 17 **EOSMP_SLV:** End of sampling phase flag of the slave ADC
This bit is a copy of the EOSMP2 bit in the corresponding ADC_ISR register.

Bit 16 **ADRDY_SLV:** Slave ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **LDORDY_MST:** ADC internal voltage regulator flag of the master ADC
This bit is a copy of the LDORDY bit in the corresponding ADC_ISR register.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **JQOVF_MST:** Injected context queue overflow flag of the master ADC
This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 9 **AWD3_MST:** Analog watchdog 3 flag of the master ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 8 **AWD2_MST:** Analog watchdog 2 flag of the master ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 7 **AWD1_MST:** Analog watchdog 1 flag of the master ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 6 **JEOS_MST:** End of injected sequence flag of the master ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 5 **JEOC_MST:** End of injected conversion flag of the master ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 4 **OVR_MST:** Overrun flag of the master ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 3 **EOS_MST:** End of regular sequence flag of the master ADC
This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 2 **EOC_MST:** End of regular conversion of the master ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 1 **EOSMP_MST:** End of Sampling phase flag of the master ADC
This bit is a copy of the EOSMP bit in the corresponding ADC_ISR register.

Bit 0 **ADRDY_MST:** Master ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

25.7.2 ADC common control register (ADCC_CCR)

Address offset: 0x008

Reset value: 0x0000 0000

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

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

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VBATEN**: VBAT enable

This bit is set and cleared by software to enable/disable the V_{BAT} channel.

0: V_{BAT} channel disabled

1: V_{BAT} channel enabled

Note: V_{BAT} is not available on all ADC instances. Refer to Section 25.4.4: ADC connectivity for details.

Bit 23 **TSEN**: Temperature sensor voltage enable

This bit is set and cleared by software to control V_{SENSE} channel.

0: Temperature sensor channel disabled

1: Temperature sensor channel enabled

Note: The temperature sensor is not available on all ADC instances. Refer to Section 25.4.4: ADC connectivity for details.

Note: The software is allowed to write this bit only when ADEN = 0.

Bit 22 **VREFEN**: V_{REFINT} enable

This bit is set and cleared by software to enable/disable the V_{REFINT} channel.

0: V_{REFINT} channel disabled

1: V_{REFINT} channel enabled

Note: V_{REFINT} is not available on all ADC instances. Refer to Section 25.4.4: ADC connectivity for details.

Note: The software is allowed to write this bit only when ADEN = 0.

Bits 21:16 Reserved, must be kept at reset value.

Bits 15:14 **DAMDF[1:0]:** Dual ADC mode data format

This bitfield are set and cleared by software. It specifies the data format in the common data register ADCC_CDR and ADCC_CDR2.

00: Dual ADC mode without data packing (ADCC_CDR and ADCC_CDR2 registers not used).

01: Reserved

10: Data formatting mode for any data width (ADCC_CDR data register is used when the data width is less than 16 bits, otherwise ADCC_CDR2 register is used)

11: Data formatting mode for data width lower that 8 bits (ADCC_CDR data register is used)

Note: The software is allowed to write these bits only when ADEN = 0 (ADC is disabled).

Bit 13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **DELAY[3:0]:** Delay between two sampling phases

These bits are set and cleared by software. These bits are used in dual interleaved modes. Refer to *Table 253* for the value of ADC resolution versus DELAY bits values.

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DUAL[4:0]:** Dual ADC mode selection

These bits are written by software to select the operating mode.
All the ADCs independent:
00000: Independent mode
The following settings apply to dual mode, master and slave ADCs working together
00001: Combined regular simultaneous + Injected simultaneous mode
00010: Combined regular simultaneous + Alternate trigger mode
00011: Combined interleaved mode + Injected simultaneous mode
00101: Injected simultaneous mode only
00110: Regular simultaneous mode only
00111: Interleaved mode only
01001: Alternate trigger mode only
Others: Reserved
All other combinations are reserved and must not be programmed

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

25.7.3 ADC common regular data register for dual mode (ADCC_CDR)

Address offset: 0x00C

Reset value: 0x0000 0000

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

Bits 31:16 **RDATA_SLV[15:0]:** Regular data of the slave ADC

In dual mode, these bits contain the regular data of the slave ADC. Refer to *Section 25.4.34: Dual ADC modes*.

The data alignment is applied as described in *Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS, JLSHIFT[a], USAT, SSAT, POSOFF)*)

Bits 15:0 **RDATA_MST[15:0]**: Regular data of the master ADC.

In dual mode, these bits contain the regular data of the master ADC. Refer to *Section 25.4.34: Dual ADC modes*.

The data alignment is applied as described in *Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS, JLSHIFT[a], USAT, SSAT, POSOFF)*)

In MDMA=0b11 mode, bits 15:8 contains SLV_ADC_DR[7:0], bits 7:0 contains MST_ADC_DR[7:0].

25.7.4 ADC common regular data register for dual mode (ADCC_CDR2)

Address offset: 0x010

Reset value: 0x0000 0000

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

Bits 31:0 **RDATA_ALT[31:0]**: Regular data of the master/slave alternated ADCs.

In dual mode, these bits contain the regular 32-bit data of the master and the slave ADC. Refer to *Section 25.4.34: Dual ADC modes*.

The data alignment is applied as described in *Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS, JLSHIFT[a], USAT, SSAT, POSOFF)*)

25.8 ADC register map

ADC1 and ADC3 are master ADCs.

ADC2 and ADC4 operate as slave from ADC1 and ADC3, respectively.

Table 255. ADC register map and reset values for each ADC

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ADC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDORDY	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
	Reset value																				0		0	0	0	0	0	0	0	0	0	0	0

Table 255. ADC register map and reset values for each ADC (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	ADC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDORDYIE	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																				0		0	0	0	0	0	0	0	0	0	0	0
0x08	ADC_CR	ADCAL	ADCALDIF	DEEPPWD	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
	Reset value	0	0	1	0																							0	0	0	0	0	0
0x00C	ADC_CFGR1	JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM[2:0]			DISCEN	Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL[4:0]					Res.	RES[1:0]		DMNGT[1:0]	
	Reset value	0x8000 0000																															
0x010	ADC_CFGR2	LSHIFT[3:0]				Res.	Res.	OVSR[9:0]										SMPTRIG	SWTRIG	BULB	Res.	Res.	ROMVSM	TROVS	OVSS[3:0]				Res.	PDIFFREF	PDIFF	Res.	ROVSE
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0		0	0		0
0x014	ADC_SMPR1	Res.	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:0]			SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]	
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	ADC_SMPR2	Res.	Res.	SMP19[2:0]			SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:0]			SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]	
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	ADC_PCSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCSEL[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020-0x02C	Reserved																																
0x030	ADC_SQR1	Res.	Res.	Res.	SQ4[4:0]				Res.	SQ3[4:0]				Res.	SQ2[4:0]				Res.	SQ1[4:0]				Res.	Res.	LEN[3:0]							
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0
0x034	ADC_SQR2	Res.	Res.	Res.	SQ9[4:0]				Res.	SQ8[4:0]				Res.	SQ7[4:0]				Res.	SQ6[4:0]				Res.	Res.	SQ5[4:0]							
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0	0
0x038	ADC_SQR3	Res.	Res.	Res.	SQ14[4:0]				Res.	SQ13[4:0]				Res.	SQ12[4:0]				Res.	SQ11[4:0]				Res.	Res.	SQ10[4:0]							
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0	0
0x03C	ADC_SQR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
	Reset value																						0	0	0	0	0		0	0	0	0	0
0x040	ADC_DR	RDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044-0x48	Reserved																																
0x04C	ADC_JSQR	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:0]				Res.	JSQ1[4:0]				JEXTEN[1:0]		JEXTSEL[4:0]					JLEN[1:0]		
	Reset value	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 255. ADC register map and reset values for each ADC (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x050	ADC_OFCFGR1			OFFSET1_CH[4:0]			SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x054	ADC_OFCFGR2			OFFSET2_CH[4:0]			SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x058	ADC_OFCFGR3			OFFSET3_CH[4:0]			SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x05C	ADC_OFCFGR4			OFFSET4_CH[4:0]			SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x060	ADC_OFR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	ADC_OFR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	ADC_OFR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	ADC_OFR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	ADC_GCOMP	GCOMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GCOMPCOEFF[13:0]													
	Reset value	0																		0	1	0	0	0	0	0	0	0	0	0	0	0	0
0x074	ADC_AWD1SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDS_CH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x078	ADC_AWD2SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDS_CH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	ADC_AWD3SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDS_CH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x080	ADC_JDR1	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	ADC_JDR2	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 255. ADC register map and reset values for each ADC (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x088	ADC_JDR3	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	ADC_JDR4	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	ADC_CFGR3	JLSHIFT[3:0]				Res.	Res.	JOVSR[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	TJOVS	JOVSS[3:0]				Res.	Res.	JOVSIVE	JOVSE	JOVSPAR
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0							0	0	0	0	0			0	0	0
0x094	ADC_JOVSCR1	Res.	Res.	Res.	Res.	Res.	Res.	JOVSC2[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	JOVSC1[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x098	ADC_JOVSCR2	Res.	Res.	Res.	Res.	Res.	Res.	JOVSC4[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	JOVSC3[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x0A0	ADC_AWD2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	ADC_AWD3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A8	ADC_AWD1LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0AC	ADC_AWD1HTR	AWDFILT [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:0]																						
	Reset value	0	0	0							0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0B0	ADC_AWD2LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4	ADC_AWD2HTR	AWDFIL [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:0]																						
	Reset value	0	0	0							0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0B8	ADC_AWD3LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BC	ADC_AWD3HTR	AWDFIL [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:0]																						
	Reset value	0	0	0							0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0C0	ADC_DIFSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C8-0x0CC	Reserved																																
0x0D0	ADC_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OP7	OP6	OP5	OP4	OP3	OP2	OP1	OP0
	Reset value																									0	0	0	0	0	0	0	0
0x0D4-0x0FC	Reserved																																

Table 256. ADC register map and reset values (master and slave ADC common registers)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ADCC_CSR	Res.	Res.	Res.	LDORDY_SLV	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV	Res.	Res.	Res.	LDORDY_MST	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
					slave ADC																master ADC												
	Reset value				0		0	0	0	0	0	0	0	0	0	0	0				0		0	0	0	0	0	0	0	0	0	0	0
0x004	Reserved	Res.																															
0x008	ADCC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	TSEN	VREFEN	Res.				Res.	Res.	DAMDF[1:0]		Res.	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
	Reset value								0	0	0							0	0			0	0	0	0				0	0	0	0	0
0x00C	ADCC_CDR	RDATA_SLV[15:0]																RDATA_MST[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	ADCC_CDR2	RDATA_ALT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	Reserved																																

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

26 Digital temperature sensor (DTS)

26.1 DTS introduction

The device embeds a sensor that converts the temperature into a square wave which frequency is proportional to the temperature. The frequency is measured either with the PCLK or the LSE clock.

26.2 DTS main features

The temperature sensor block main features are the following:

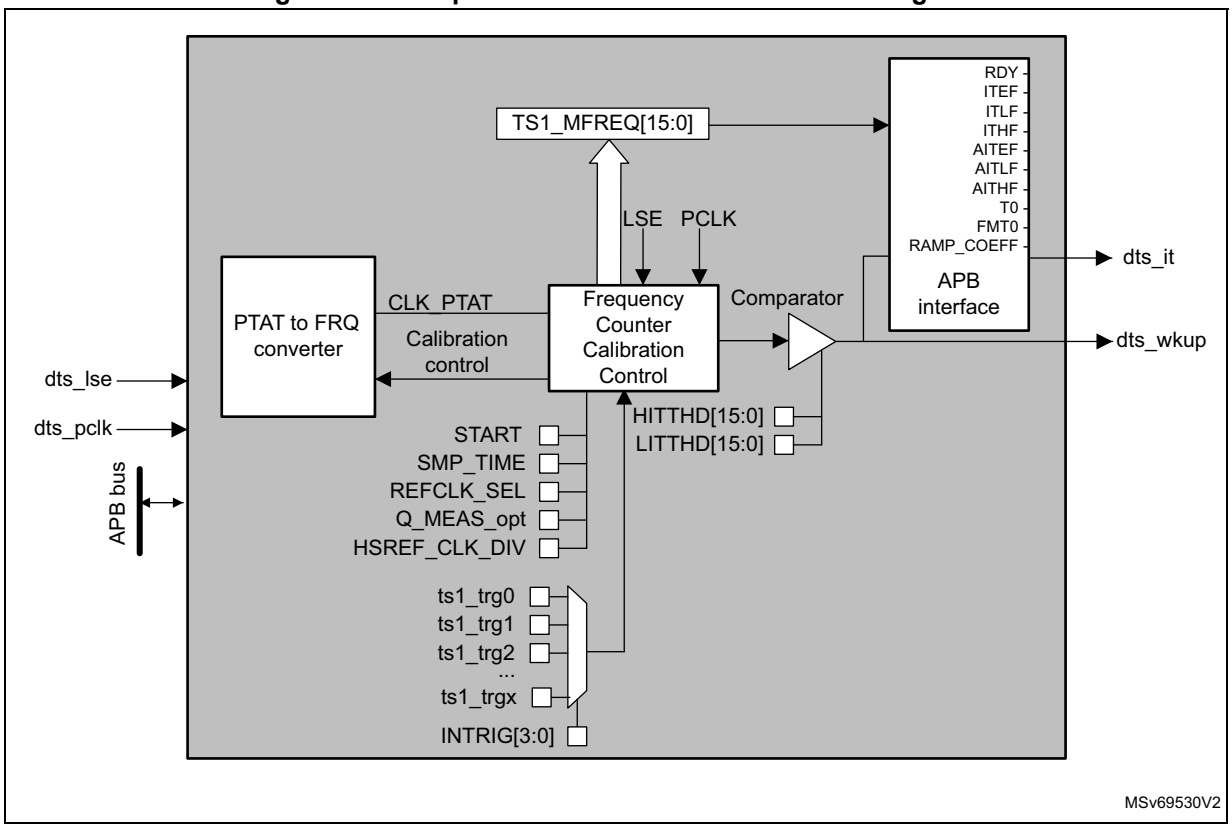
- Start of measurement triggered by software or 4 hardware sources
- Programmable sampling time to increase temperature measurement precision
- Counter synchronized on LSE or PCLK clock
- Temperature watchdog on low and high threshold
- Interrupt generation when the temperature is lower or higher than predefined thresholds and at the end of measurement.
- Asynchronous wake-up signal generation when the temperature is higher/lower than a predefined threshold (LSE mode only)
- Quick measurement using LSE clock

26.3 DTS functional description

26.3.1 DTS block diagram

The temperature sensor block diagram is shown in *Figure 358*.

Figure 358. Temperature sensor functional block diagram



26.3.2 DTS internal signals

Table 257. DTS internal input/output signals

Signal name	Signal type	Description
dts_lse	Digital input	LSE clock
dts_pclk	Digital input	APB clock
dts_it	Digital output	Temperature sensor interrupt
dts_wkup	Digital output	Temperature sensor wake-up

26.3.3 DTS block operation

The analog part of the temperature sensor outputs a frequency that is proportional to the absolute temperature (CLK_PTAT). The frequency measurement is based on the PCLK or the LSE clock.

Before each measurement, the temperature sensor performs a calibration of the frequency generation blocks.

26.3.4 Operating modes

Several operating modes can be selected by setting the REFCLK_SEL bit in *Temperature sensor configuration register 1 (DTS_CFGR1)*:

- PCLK only (REFCLK_SEL = 0)

  The temperature sensor registers can be accessed. The interface can consequently be reconfigured and the measurement sequence is performed using PCLK clock
- PCLK and LSE (REFCLK_SEL = 1)

  The temperature sensor registers can be accessed. The interface can consequently be reconfigured and the measurement sequence is performed using the LSE clock.
- LSE only (REFCLK_SEL = 1) and PCLK OFF

  The registers cannot be accessed. The measurement can be performed using the LSE clock. This mode is used to exit from Sleep mode by using hardware triggers and the asynchronous interrupt line.

26.3.5 Calibration

The temperature sensor must run the calibration prior to any frequency measurement. The calibration is performed automatically when the temperature measurement is triggered except for quick measurement mode (Q_MEAS_OPT set to 1 in DTS_CFGR1).

26.3.6 Prescaler

When a calibration is ongoing, the counter clock must be slower than 1 MHz. This is achieved by the PCLK clock prescaler embedded in the temperature sensor.

During the temperature measurement period, the prescaler is bypassed.

- When PCLK is used as reference clock (REFCLK_SEL set to 0 in DTS_CFGR1), a prescaler is used. Its division ratio must be configured up to 127 (refer to the HSREF_CLK_DIV[6:0] register definition for the divider setting).
- When LSE is used as reference clock (REFCLK_SEL set to 1 in DTS_CFGR1), the timebase is equal to 2 LSE periods. In this case, no prescaler is used.

26.3.7 Temperature measurement principles

The analog part of temperature sensor outputs a signal (CLK_PTAT) which FM(T) frequency is temperature-dependent.

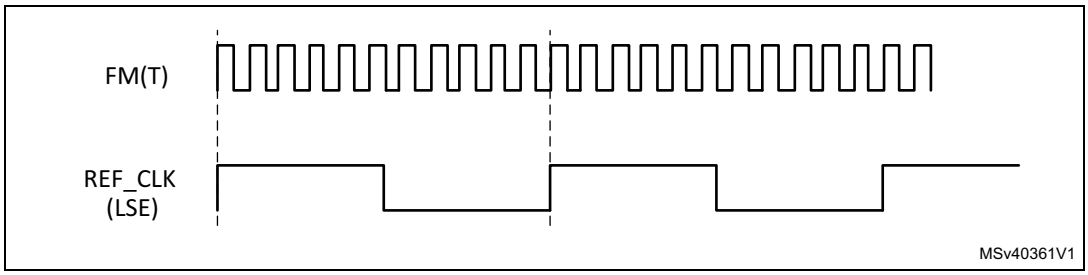
Either PCLK or LSE can be selected as reference clock (REF_CLK) through the REFCLK_SEL bit in DTS_CFGR1.

The counting method depends on the REF_CLK frequency. This is due to the fact that two counters are implemented in the temperature sensor block:

- For low REF_CLK frequencies, a counting of FM(T) cycles is performed during one or several REF_CLK cycles.
- For high REF_CLK frequencies, a counting of REF_CLK cycles is performed during one or several FM(T) cycles.

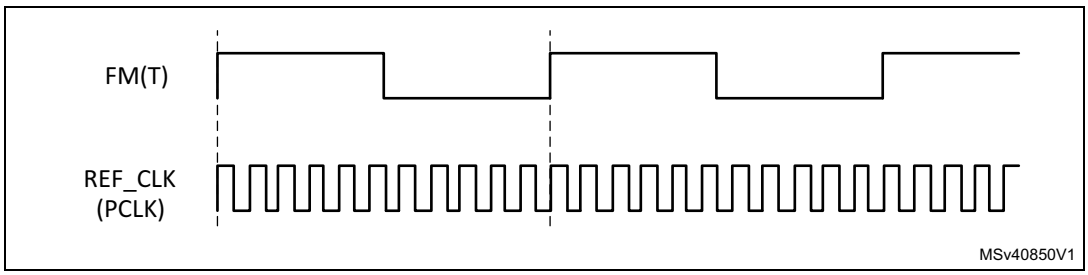
This counter behavior is shown in *Figure 359* and *Figure 360*.

Figure 359. Method for low REF_CLK frequencies



1. To increase the precision, FM(T) measurement can be done on several LSE periods.

Figure 360. Method for high REF_CLK frequencies



1. To increase the precision, PCLK measurement can be done on several FM(T) periods.

The counting result is stored in the DTS_DR register (see *Temperature sensor data register (DTS_DR)*.

Once the FM(T) frequency has been obtained, the corresponding temperature can be calculated by software using the following formula:

- When PCLK is used:

$$T = T_0 + ((F_{PCLK} / TS1_MFREQ) \times TS1_SMP_TIME - 100 \times TS1_FMT0) / TS1_RAMP_COEFF$$

where

T0 (factory calibration temperature) is equal to 30 °C.

TS1_FMT0 is measured and stored in the DTS_T0VALR1 register. It is expressed in hundreds of Hertz.

TS1_RAMP_COEFF is measured during tests in factory and stored in DTS_RAMPVALR register. This value is expressed in Hz/°C.

- When the LSE clock is used

$$T = T_0 + ((F_{LSE} \times TS1_MFREQ / TS1_SMP_TIME) - (100 \times TS1_FMT0)) / TS1_RAMP_COEFF$$

26.3.8 Sampling time

The sampling period can be increased to improve measurement accuracy. This is useful when the reference frequency (REF_CLK) is close to the FM(T) frequency. The default value is one REF_CLK cycle in LSE mode, and one FM(T) cycle in PCLK mode.

The sampling time is configured through TS1_SMP_TIME bits in DTS_CFGR1 register (see *Table 258*).

Table 258. Sampling time configuration

TS1_SMP_TIME[3:0]	LSE or FM(T) clock cycle(s)
0000	1
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

26.3.9 Quick measurement mode

If a high precision is not required, the calibration step included in each measurement sequence can be skipped by setting Q_MEAS_OPT to 1 in the DTS_CFGR1 register. This method must used only when the LSE clock is selected as reference clock (LSREF_CLK set to 1). This mode can reduce the measurement time.

26.3.10 Trigger input

The temperature measurement can be triggered either by software or by an external event. The trigger source can be selected through TS1_INTRIG[3:0] bits in DTS_CFGR1.

Table 259. Trigger configuration

Name	TS1_INTRIG[3:0]				Comment
N.A	0	0	0	0	No hardware trigger
ts1_trg0	0	0	0	1	lptim4_out
ts1_trg1	0	0	1	0	lptim2_ch1
ts1_trg2	0	0	1	1	lptim3_ch1
ts1_trg3	0	1	0	0	exti13
ts1_trg4	0	1	0	1	Reserved
ts1_trg5	0	1	1	0	
ts1_trg6	0	1	1	1	
ts1_trg7	1	0	0	0	
ts1_trg8	1	0	0	1	
ts1_trg9	1	0	1	0	
ts1_trg10	1	0	1	1	
ts1_trg11	1	1	0	0	
ts1_trg12	1	1	0	1	
ts1_trg13	1	1	1	0	
ts1_trg14	1	1	1	1	

Note: Hardware triggers are active only on the rising edge.

The temperature sensor can only capture a hardware trigger rising edge when TS1_RDY bit is set (see Section 26.3.11: On-off control and ready flag), otherwise the trigger is ignored.

If a trigger source changes on-the-fly, the new trigger source signal must be low. If the new source signal is high, the temperature sensor detects a rising edge and start the measurement sequence.

26.3.11 On-off control and ready flag

The DTS block can be enabled by setting TS1_EN bit in DTS_CFGR1 register. The TS1_RDY flag in the *Temperature sensor status register (DTS_SR)* indicate that the DTS block is ready for temperature measurement: when TS1_RDY bit is set to 1, the measurement can be started. Once a measurement has started, TS1_RDY bit is reset. New measurement requests are then ignored. Once the measurement is finished, TS1_RDY bit is set again to indicate the sensor is ready to start a new measurement.

26.3.12 Temperature measurement sequence

Start of measurement can be triggered by software or hardware.

Software trigger

The software trigger is selected when TS1_INTRIG_SEL[3:0] is set to ‘0000’ in DTS_CFGR1.

If TS1_RDY is set to 1, writing TS1_START bit to 1 in DTS_CFGR1 starts the measurement.

If TS1_RDY equals 0, the software trigger does not start until TS1_RDY is set.

If TS1_START bit is kept at 1 once the measurement is finished, then the TS1_RDY flag become 1 and the measurement restarts.

Hardware trigger

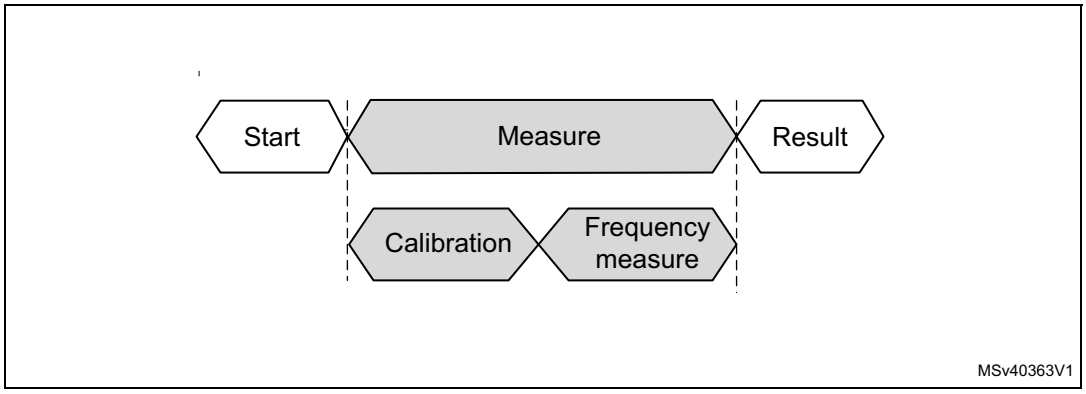
TS1_INTRIG_SEL[3:0] bits allow selecting one hardware trigger out of 4. If TS1_RDY is set to 1, a rising edge on the trigger signal starts the measurement. When TS1_RDY is 0, the rising edge is ignored.

Temperature measurement sequence

One measurement contains two steps: the calibration of the analog blocks and the measurement. The calibration automatically starts when the measurement is triggered (see *Section 26.3.5: Calibration*). The measurement period depends on the following DTS_CFGR1 bits:

- the reference clock selected through REFCLK_SEL bit
- the divider ratio configured by HSREF_CLK_DIV bits
- the sampling time defined by TS1_SMP_TIME bits.

Figure 361. Temperature sensor sequence



26.4 DTS low-power modes

Table 260. Temperature sensor behavior in low-power modes

Mode	Description
Sleep	Only works in LSE mode. DTS interrupt causes the device to exit from Sleep mode.
Stop	Only works in LSE mode. DTS interrupt cause the device to exit from Stop mode.

26.5 DTS interrupts

There are two ways to use the DTS block as an interrupt source.The DTS interrupt line can be connected to the EXTI controller (see *Section 26.5.3: Asynchronous wake-up*) or to the CPU NVIC (see *Section 26.5.2: Synchronous interrupt*).

26.5.1 Temperature window comparator

The DTS_ITR1 register allows defining the high and low threshold that are used for temperature comparison. If the temperature data is equal or higher than TS1_HITTHD, or equal or lower than TS1_LITTHD bit, an interrupt is generated and the corresponding flag, TS1_ ITLF, TS1_ITHF, TS1_AITLF and TS1_AITHF, is set in the DTS_SR register (see *Section 26.6.6*).

26.5.2 Synchronous interrupt

A global interrupt output line is available on the DTS block. The interrupt can be generated at the end of measurement and/or when the measurement result is equal/higher or equal/lower than a predefined threshold (see *Section 26.5.1: Temperature window comparator*).

Three interrupt events can be select via 3 bits in DTS_ITENR register (see *Section 26.6.7*). All combinations of interrupts are allowed.

The TS1_ITEF, TS1_ITLF and TS1_ITHF flags in the DTS_SR register reflect the interrupt event. They can be reset with the correspond bits of the DTS_ICIFR register (see *Section 26.6.8*).

26.5.3 Asynchronous wake-up

The DTS block also provides an asynchronous interrupt line. It is used only when the LSE is selected as reference clock (REFCLK_SEL=1).

This line can generate a signal that wakes up the system from Sleep mode at the end of measurement and/or when the measurement result is equal/higher or equal/lower than a predefined threshold (see *Section 26.5.1: Temperature window comparator*).

Three asynchronous wake-up events can be selected via 3 bits in DTS_ITENR register. All combination of interrupts are allowed.

The TS1_AITEF, TS1_AITLF and TS1_AITHF flags in the DTS_SR register reflect the interrupt status. They can be reset with the correspond bits of the DTS_ICIFR register.

The following table shows the interrupt bits an the their description.

26.6 DTS registers

The registers of this peripheral can only be accessed by-word (32-bit).

26.6.1 Temperature sensor configuration register 1 (DTS_CFGR1)

DTS_CFGR1 is the configuration register for temperature sensor 1.

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	HSREF_CLK_DIV[6:0]							Res.	Res.	Q_MEAS_OPT	REFCLK_SEL	TS1_SMP_TIME[3:0]			
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TS1_INTRIG_SEL[3:0]				Res.	Res.	Res.	TS1_START	Res.	Res.	Res.	TS1_EN
				rw	rw	rw	rw				rw				rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **HSREF_CLK_DIV[6:0]**: High speed clock division ratio

These bits are set and cleared by software. They can be used to define the division ratio for the main clock in order to obtain the internal frequency lower than 1 MHz required for the calibration. They are applicable only for calibration when PCLK is selected as reference clock (REFCLK_SEL=0).

0000000: No divider

0000001: No divider

0000010: 1/2 division ratio

...

1111111: 1/127 division ratio

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **Q_MEAS_OPT**: Quick measurement option bit

This bit is set and cleared by software. It is used to increase the measurement speed by suppressing the calibration step. It is effective only when the LSE clock is used as reference clock (REFCLK_SEL=1).

0: Measurement with calibration

1: Measurement without calibration

Bit 20 **REFCLK_SEL**: Reference clock selection bit

This bit is set and cleared by software. It indicates whether the reference clock is the high speed clock (PCLK) or the low speed clock (LSE).

0: High speed reference clock (PCLK)

1: Low speed reference clock (LSE)

Bits 19:16 **TS1_SMP_TIME[3:0]**: Sampling time for temperature sensor 1
These bits allow increasing the sampling time to improve measurement precision.
When the PCLK clock is selected as reference clock (REFCLK_SEL = 0), the measurement is performed at TS1_SMP_TIME period of CLK_PTAT.
When the LSE is selected as reference clock (REFCLK_SEL = 1), the measurement is performed at TS1_SMP_TIME period of LSE.

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

Bits 11:8 **TS1_INTRIG_SEL[3:0]**: Input trigger selection bit for temperature sensor 1
These bits are set and cleared by software. They select which input triggers a temperature measurement. Refer to *Section 26.3.10: Trigger input*.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **TS1_START:** Start frequency measurement on temperature sensor 1
This bit is set and cleared by software.
0: No software trigger.
1: Software trigger for a frequency measurement. (only if TS1 is ready).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **TS1_EN:** Temperature sensor 1 enable bit
This bit is set and cleared by software.
0: Temperature sensor 1 disabled
1: Temperature sensor 1 enabled

Note: Once enabled, the temperature sensor is active after a specific delay time. The TS1_RDY flag is set when the sensor is ready.

26.6.2 Temperature sensor T0 value register 1 (DTS_T0VALR1)

DTS_T0VALR1 contains the value of the factory calibration temperature (T0) for temperature sensor 1. The reset value is factory trimmed.

Address offset: 0x08

Reset value: 0x000X XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_T0[1:0]	
														r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_FMT0[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 17:16 **TS1_T0[1:0]**: Engineering value of the T0 temperature for temperature sensor 1.
00: 30 °C
01: 140 °C
Others: Reserved, must not be used.

Bits 15:0 **TS1_FMT0[15:0]**: Engineering value of the frequency measured at T0 for temperature sensor 1
This value is expressed in 0.1 kHz.

26.6.3 Temperature sensor ramp value register (DTS_RAMPVALR)

The DTS_RAMPVALR is the ramp coefficient for the temperature sensor. The reset value is factory trimmed.

Address offset: 0x10

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_RAMP_COEFF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TS1_RAMP_COEFF[15:0]**: Engineering value of the ramp coefficient for the temperature sensor 1.

This value is expressed in Hz/°C.

26.6.4 Temperature sensor interrupt threshold register 1 (DTS_ITR1)

DTS_ITR1 contains the threshold values for sensor 1.

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TS1_HITTHD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_LITTHD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **TS1_HITTHD[15:0]:** High interrupt threshold for temperature sensor 1

These bits are set and cleared by software. They indicate the highest value than can be reached before raising an interrupt signal.

Bits 15:0 **TS1_LITTHD[15:0]:** Low interrupt threshold for temperature sensor 1

These bits are set and cleared by software. They indicate the lowest value than can be reached before raising an interrupt signal.

26.6.5 Temperature sensor data register (DTS_DR)

The DTS_DR contains the number of REF_CLK cycles used to compute the FM(T) frequency.

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_MFREQ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TS1_MFREQ[15:0]**: Value of the counter output value for temperature sensor 1

26.6.6 Temperature sensor status register (DTS_SR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_ RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_ AITHF	TS1_ AITLF	TS1_ AITEF	Res.	TS1_ ITHF	TS1_ ITLF	TS1_ ITEF
r									r	r	r		r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TS1_RDY:** Temperature sensor 1 ready flag
This bit is set and reset by hardware.
It indicates that a measurement is ongoing.
0: Temperature sensor 1 busy
1: Temperature sensor 1 ready

Bits 14:7 Reserved, must be kept at reset value.

Bit 6 **TS1_AITHF:** Asynchronous interrupt flag for high threshold on temperature sensor 1
This bit is set by hardware when the high threshold is reached.
It is cleared by software by writing 1 to the TS1_CAITHF bit in the DTS_ICIFR register.
0: High threshold not reached on temperature sensor 1
1: High threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_AITHFEN bit is set

Bit 5 **TS1_AITLF:** Asynchronous interrupt flag for low threshold on temperature sensor 1
This bit is set by hardware when the low threshold is reached.
It is cleared by software by writing 1 to the TS1_CAITLF bit in the DTS_ICIFR register.
0: Low threshold not reached on temperature sensor 1
1: Low threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_AITLFEN bit is set

Bit 4 **TS1_AITEF:** Asynchronous interrupt flag for end of measure on temperature sensor 1
This bit is set by hardware when a temperature measure is done.
It is cleared by software by writing 1 to the TS1_CAITEF bit in the DTS_ICIFR register.
0: End of measure not detected on temperature sensor 1
1: End of measure detected on temperature sensor 1
Note: This bit is active only when the TS1_AITEFEN bit is set

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_ITHF:** Interrupt flag for high threshold on temperature sensor 1, synchronized on PCLK
This bit is set by hardware when the high threshold is set and reached.
It is cleared by software by writing 1 to the TS1_CITHF bit in the DTS_ICIFR register.
0: High threshold not reached on temperature sensor 1
1: High threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_ITHFEN bit is set

Bit 1 **TS1_ITLF:** Interrupt flag for low threshold on temperature sensor 1, synchronized on PCLK.
This bit is set by hardware when the low threshold is set and reached.
It is cleared by software by writing 1 to the TS1_CITLF bit in the DTS_ICIFR register.
0: Low threshold not reached on temperature sensor 1
1: Low threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_ITLFEN bit is set

Bit 0 **TS1_ITEF:** Interrupt flag for end of measurement on temperature sensor 1, synchronized on PCLK.
This bit is set by hardware when a temperature measure is done.
It is cleared by software by writing 1 to the TS2_CITEF bit in the DTS_ICIFR register.
0: No end of measurement detected on temperature sensor 1
1: End of measure detected on temperature sensor 1
Note: This bit is active only when the TS1_ITEFEN bit is set

26.6.7 Temperature sensor interrupt enable register (DTS_ITENR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_AITHEN	TS1_AITLEN	TS1_AITEEN	Res.	TS1_ITHEN	TS1_ITLEN	TS1_ITEEN
									rw	rw	rw		rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **TS1_AITHEN:** Asynchronous interrupt enable flag on high threshold for temperature sensor 1.
This bit are set and cleared by software.
It enables the asynchronous interrupt when the temperature is above the high threshold (only when REFCLK_SEL= 1")
0: Asynchronous interrupt on high threshold disabled for temperature sensor 1
1: Asynchronous interrupt on high threshold enabled for temperature sensor 1

Bit 5 **TS1_AITLEN:** Asynchronous interrupt enable flag for low threshold on temperature sensor 1.
This bit are set and cleared by software.
It enables the asynchronous interrupt when the temperature is below the low threshold (only when REFCLK_SEL= 1)
0: Asynchronous interrupt on low threshold disabled for temperature sensor 1
1: Asynchronous interrupt on low threshold enabled for temperature sensor 1

Bit 4 **TS1_AITEEN:** Asynchronous interrupt enable flag for end of measurement on temperature sensor 1
This bit are set and cleared by software.
It enables the asynchronous interrupt for end of measurement (only when REFCLK_SEL = 1).
0: Asynchronous interrupt for end of measurement disabled on temperature sensor 1
1: Asynchronous interrupt for end of measurement enabled on temperature sensor 1

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_ITHEN:** Interrupt enable flag for high threshold on temperature sensor 1, synchronized on PCLK.
This bit are set and cleared by software.
It enables the interrupt when the measure reaches or is above the high threshold.
0: Synchronous interrupt for high threshold disabled on temperature sensor 1
1: Synchronous interrupt for high threshold enabled on temperature sensor 1

Bit 1 **TS1_ITLEN:** Interrupt enable flag for low threshold on temperature sensor 1, synchronized on PCLK.
This bit are set and cleared by software.
It enables the synchronous interrupt when the measure reaches or is below the low threshold.
0: Synchronous interrupt for low threshold disabled on temperature sensor 1
1: Synchronous interrupt for low threshold enabled on temperature sensor 1

Bit 0 **TS1_ITEEN:** Interrupt enable flag for end of measurement on temperature sensor 1, synchronized on PCLK.
This bit are set and cleared by software.
It enables the synchronous interrupt for end of measurement.
0: Synchronous interrupt for end of measurement disabled on temperature sensor 1
1: Synchronous interrupt for end of measurement enabled on temperature sensor 1

26.6.8 Temperature sensor clear interrupt flag register (DTS_ICIFR)

DTS_ICIFR is the control register for the interrupt flags.

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_ CAITHF	TS1_ CAITLF	TS1_ CAITEF	Res.	TS1_ CITHF	TS1_ CITLF	TS1_ CITEF
									rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **TS1_CAITHF:** Asynchronous interrupt clear flag for high threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_AITHF flag in the DTS_SR register.

Bit 5 **TS1_CAITLF:** Asynchronous interrupt clear flag for low threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_AITLF flag in the DTS_SR register.

Bit 4 **TS1_CAITEF:** Write once bit. Clear the asynchronous IT flag for End Of Measure for thermal sensor 1.
Writing 1 clears the TS1_AITEF flag of the DTS_SR register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_CITHF:** Interrupt clear flag for high threshold on temperature sensor 1
Writing this bit to 1 clears the TS1_ITHF flag in the DTS_SR register.

Bit 1 **TS1_CITLF:** Interrupt clear flag for low threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_ITLF flag in the DTS_SR register.

Bit 0 **TS1_CITEF:** Interrupt clear flag for end of measurement on temperature sensor 1
Writing 1 to this bit clears the TS1_ITEF flag in the DTS_SR register.

26.6.9 Temperature sensor option register (DTS_OR)

The DTS_OR contains general-purpose option bits.

Address offset: 0x2C

Reset value: 0x0000 0000

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

Bits 31:0 **TS_OP[31:0]:** general purpose option bits

26.6.10 DTS register map

The following table summarizes the temperature sensor registers.

Table 261. DTS register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DTS_ CFGR1	Res.	HSREF_CLK_DIV [6:0]							Res.	Res.	Q_MEAS_OPT	REFCLK_SEL	TS1_ SMP_TIME [3:0]				Res.	Res.	Res.	Res.	TS1_ INTRIG_SEL [3:0]				Res.	Res.	Res.	TS1_START	Res.	Res.	Res.	TS1_EN
	Reset value		0	0	0	0	0	0	0			0	0	0	0	0	0					0	0	0	0				0				0
0x04	Reserved																																
0x08	DTS_ T0VALR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_T0[1:0]		TS1_FMT0[15:0]															
	Reset value															X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0C	Reserved																																
0x10	DTS_ RAMPVALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_RAMP_COEFF[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x14	DTS_ITR1	TS1_HITTHD[15:0]																TS1_LITTHD[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	Reserved																																
0x1C	DTS_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_MFREQ[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	DTS_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_AITHF	TS1_AITLF	TS1_AITEF	Res.	TS1_ITHF	TS1_ITLF	TS1_ITEF
	Reset value																	0									0	0	0		0	0	0
0x24	DTS_ITENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_AITHEN	TS1_AITLEN	TS1_AITEEN	Res.	TS1_ITHEN	TS1_ITLEN	TS1_ITEEN
	Reset value																										0	0	0		0	0	0
0x28	DTS_ICIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_CAITHF	TS1_CAITLF	TS1_CAITEF	Res.	TS1_CITHF	TS1_CITLF	TS1_CITEF
	Reset value																										0	0	0		0	0	0

Table 261. DTS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	DTS_OR	TS_OP31	TS_OP30	TS_OP29	TS_OP28	TS_OP27	TS_OP26	TS_OP25	TS_OP24	TS_OP23	TS_OP22	TS_OP21	TS_OP20	TS_OP19	TS_OP18	TS_OP17	TS_OP16	TS_OP15	TS_OP14	TS_OP13	TS_OP12	TS_OP11	TS_OP10	TS_OP9	TS_OP8	TS_OP7	TS_OP6	TS_OP5	TS_OP4	TS_OP3	TS_OP2	TS_OP1	TS_OP0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

27 Digital-to-analog converter (DAC)

27.1 DAC introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data can be left- or right-aligned. The DAC features two output channels, each with its own converter. In dual DAC channel mode, conversions can be done independently or simultaneously when both channels are grouped together for synchronous update operations. Two input reference pins, VREFDAC+ and VREFDAC-, are available for better resolution.

The DACx_OUTy pin can be used as general purpose input/output (GPIO) when the DAC output is disconnected from output pad and connected to on chip peripheral. The DAC output buffer can be optionally enabled to obtain a high drive output current. An individual calibration can be applied on each DAC output channel. The DAC output channels support a low power mode, the sample and hold mode.

27.2 DAC main features

The DAC main features are the following (see *Figure 362: Dual-channel DAC block diagram*):

- Up to four DAC interfaces, maximum two output channels each
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Sawtooth wave generation
- Extended sawtooth wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel including DMA underrun error detection
- Double data DMA capability to reduce the bus activity
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- Each DAC output can be disconnected from the DACx_OUTy output pin
- DAC output connection to on-chip peripherals
- Sample and hold mode for low power operation in Stop mode
- Input voltage reference from VREFDAC+ and VREFDAC- pins

Figure 362 shows the block diagram of a DAC channel and *Table 263* gives the pin description.

27.3 DAC implementation

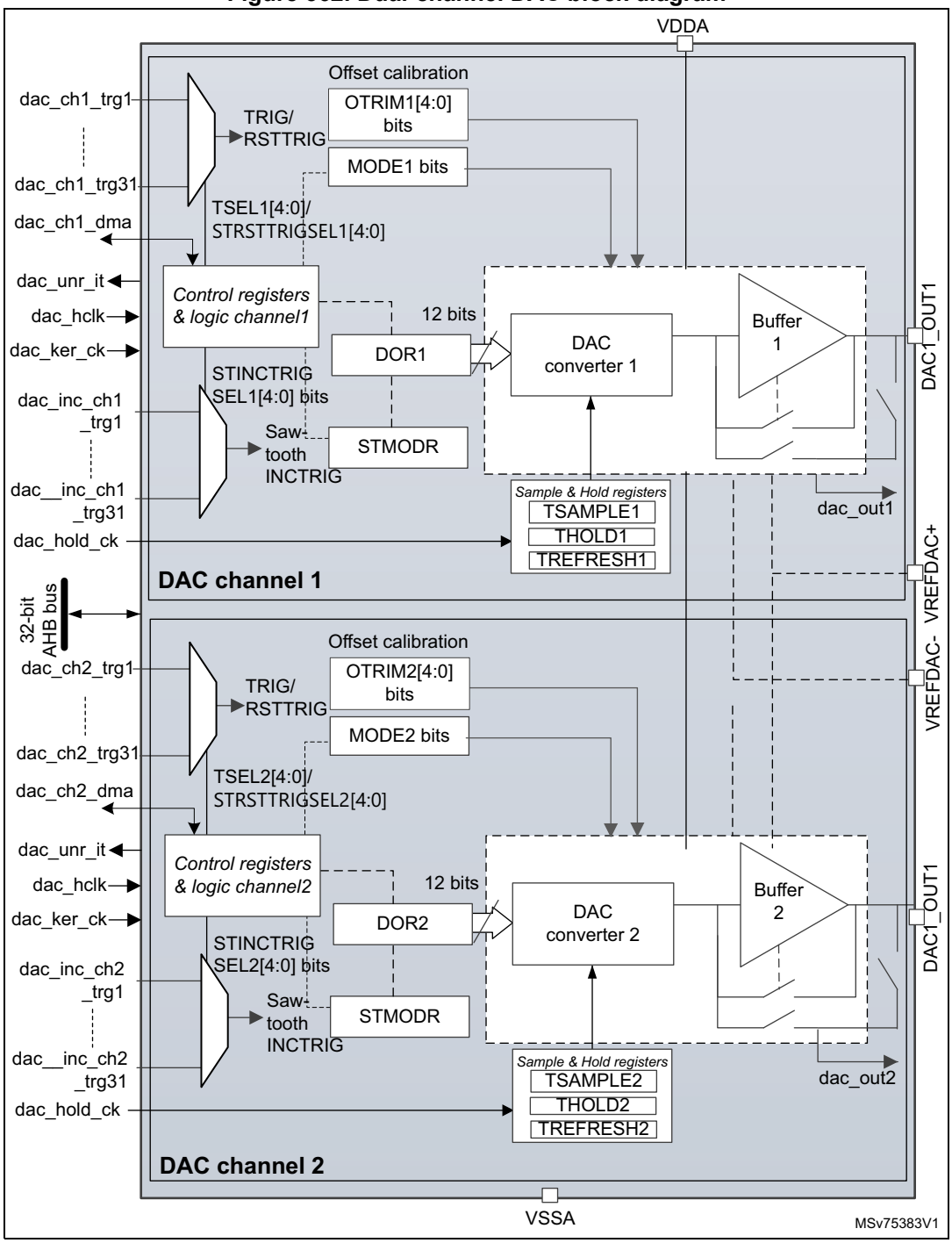
Table 262. DAC features

DAC features	DAC1	DAC2/3/4
Dual channel	-	
Output buffer	X	-
I/O connection	DAC1_OUT1 on PA4 DAC1_OUT2 on PA5	-
Maximum sampling time	1 Msps	15 Msps
Autonomous mode	-	
VREFDAC+ pin	X	
Extended sawtooth wave generation	X	
Sample and hold mode	X	-
Option register	-	X

27.4 DAC functional description

27.4.1 DAC block diagram

Figure 362. Dual-channel DAC block diagram



1. MODEx bits in the DAC_MCR control the output mode and allow switching between the normal mode in buffer/unbuffered configuration and the sample and hold mode.
2. DAC1_OUT1 and DAC1_OUT2 are available only on DAC1.

27.4.2 DAC pins and internal signals

The DAC includes:

- Up to two output channels
- The DACx_OUTy can be disconnected from the output pin and used as an ordinary GPIO
- The dac_outx can use an internal pin connection to on-chip peripherals such as comparator, operational amplifier and ADC (if available).
- DAC output channel buffered or non buffered
- Sample and hold block and registers operational in Stop mode, using the LSI/LSE clock source (dac_hold_ck) for static conversion.

The DAC includes up to two separate output channels. Each output channel can be connected to on-chip peripherals such as comparator, operational amplifier and ADC (if available). In this case, the DAC output channel can be disconnected from the DACx_OUTy output pin and the corresponding GPIO can be used for another purpose.

The DAC output can be buffered or not. The sample and hold block and its associated registers can run in Stop mode using the LSI/LSE clock source (dac_hold_ck).

Table 263. DAC input/output pins

Pin name	Signal type	Remarks
VREFDAC+	Input, analog positive reference	The higher/positive reference voltage for the DAC, $V_{REFDAC+} \leq V_{DDAmax}$ (refer to the product datasheet)
VREFDAC-	Input, analog negative reference	The lower/negative reference voltage for the DAC.
DACx_OUTy	Analog output signal	DACx channely analog output

Table 264. DAC internal input/output signals

Internal signal name	Signal type	Description
dac_ch1_dma	Bidirectional	DAC channel1 DMA request/acknowledge
dac_ch2_dma	Bidirectional	DAC channel2 DMA request/acknowledge
dac_ch1_trgx (x = 1 to 31)	Inputs	DAC channel1 trigger inputs
dac_ch2_trgx (x = 1 to 31)	Inputs	DAC channel2 trigger inputs
dac_unr_it	Output	DAC underrun interrupt
dac_hclk	Input	DAC peripheral clock
dac_ker_ck	Input	DAC kernel clock
dac_hold_ck	Input	DAC low-power clock used in sample and hold mode
dac_out1	Analog output	DAC channel1 output for on-chip peripherals
dac_out2	Analog output	DAC channel2 output for on-chip peripherals

Table 265. DAC interconnection

Signal name	Source	Source type
dac_chx_trg1	tim1_trgo	Internal signal from on-chip timer
dac_chx_trg2	tim6_trgo	Internal signal from on-chip timer
dac_chx_trg3	tim7_trgo	Internal signal from on-chip timer
dac_chx_trg4	tim8_trgo	Internal signal from on-chip timer
dac_chx_trg5	tim9_trgo	Internal signal from on-chip timer
dac_chx_trg6	tim12_trgo	Internal signal from on-chip timer
dac_chx_trg7	tim15_trgo	Internal signal from on-chip timer
dac_chx_trg8	tim20_trgo	Internal signal from on-chip timer
dac_chx_trg9	Reserved	-
dac_chx_trg10	Reserved	-
dac_chx_trg11	lptim1_ch1	Internal signal from on-chip timer
dac_chx_trg12	exti9	External pin
dac_chx_trg13	Reserved	-
dac_chx_trg14	hrtim1_dac_trg1	Internal signal from on-chip timer
dac_chx_trg15	hrtim1_dac_trg2	Internal signal from on-chip timer
dac_chx_trg16	htrim1_dac_trg3	Internal signal from on-chip timer
dac_chx_trg17	hrtim1_dac_rst_trg1	Internal signal from on-chip timer
dac_chx_trg18	hrtim1_dac_rst_trg2	Internal signal from on-chip timer
dac_chx_trg19	hrtim1_dac_rst_trg3	Internal signal from on-chip timer
dac_chx_trg20	hrtim1_dac_rst_trg4	Internal signal from on-chip timer
dac_chx_trg21	hrtim1_dac_rst_trg5	Internal signal from on-chip timer
dac_chx_trg22	hrtim1_dac_rst_trg6	Internal signal from on-chip timer
dac_chx_trg23	hrtim2_dac_rst_trg1	Internal signal from on-chip timer
dac_chx_trg24	hrtim2_dac_rst_trg2	Internal signal from on-chip timer
dac_chx_trg25	hrtim2_dac_rst_trg3	Internal signal from on-chip timer
dac_chx_trg26	hrtim2_dac_rst_trg4	Internal signal from on-chip timer
dac_chx_trg27	hrtim2_dac_rst_trg5	Internal signal from on-chip timer
dac_chx_trg28	hrtim2_dac_rst_trg6	Internal signal from on-chip timer
dac_chx_trg29	hrtim2_dac_trg1	Internal signal from on-chip timer
dac_chx_trg30	hrtim2_dac_trg2	Internal signal from on-chip timer
dac_chx_trg31	hrtim2_dac_trg3	Internal signal from on-chip timer
dac_inc_chx_trg1	tim1_trgo	Internal signal from on-chip timer
dac_inc_chx_trg2	tim1_trgo2	Internal signal from on-chip timer
dac_inc_chx_trg3	tim2_trgo (DAC1, 3) tim4_trgo (DAC2, 4)	Internal signal from on-chip timer

Table 265. DAC interconnection (continued)

Signal name	Source	Source type
dac_inc_chx_trg4	tim3_trgo (DAC1, 3) tim5_trgo (DAC2, 4)	Internal signal from on-chip timer
dac_inc_chx_trg5	tim6_trgo	Internal signal from on-chip timer
dac_inc_chx_trg6	tim7_trgo	Internal signal from on-chip timer
dac_inc_chx_trg7	tim8_trgo	Internal signal from on-chip timer
dac_inc_chx_trg8	tim8_trgo2	Internal signal from on-chip timer
dac_inc_chx_trg9	tim9_trgo	Internal signal from on-chip timer
dac_inc_chx_trg10	tim12_trgo	Internal signal from on-chip timer
dac_inc_chx_trg11	tim15_trgo	Internal signal from on-chip timer
dac_inc_chx_trg12	tim20_trgo	Internal signal from on-chip timer
dac_inc_chx_trg13	tim20_trgo2	Internal signal from on-chip timer
dac_inc_chx_trg14	Reserved	-
dac_inc_chx_trg15	Reserved	-
dac_inc_chx_trg16	lptim1_ch1	Internal signal from on-chip timer
dac_inc_chx_trg17	Reserved	-
dac_inc_chx_trg18	exti10	External pin
dac_inc_chx_trg19	Reserved	-
dac_inc_chx_trg20	hrtim1_dac_step_trg1	Internal signal from on-chip timer
dac_inc_chx_trg21	hrtim1_dac_step_trg2	Internal signal from on-chip timer
dac_inc_chx_trg22	hrtim1_dac_step_trg3	Internal signal from on-chip timer
dac_inc_chx_trg23	hrtim1_dac_step_trg4	Internal signal from on-chip timer
dac_inc_chx_trg24	hrtim1_dac_step_trg5	Internal signal from on-chip timer
dac_inc_chx_trg25	hrtim1_dac_step_trg6	Internal signal from on-chip timer
dac_inc_chx_trg26	hrtim2_dac_step_trg1	Internal signal from on-chip timer
dac_inc_chx_trg27	hrtim2_dac_step_trg2	Internal signal from on-chip timer
dac_inc_chx_trg28	hrtim2_dac_step_trg3	Internal signal from on-chip timer
dac_inc_chx_trg29	hrtim2_dac_step_trg4	Internal signal from on-chip timer
dac_inc_chx_trg30	hrtim2_dac_step_trg5	Internal signal from on-chip timer
dac_inc_chx_trg31	hrtim2_dac_step_trg6	Internal signal from on-chip timer

27.4.3 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a t_{WAKEUP} startup time.

DACxRDY bit is set in the DAC_SR register when the DAC interface is ready to accept data. Writing new data or asserting the trigger is not allowed when ENx bit is set while DACxRDY signal is reset.

Note: The ENx bit enables the analog DAC channelx only. The DAC channelx digital interface is enabled even if the ENx bit is reset.

27.4.4 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channel

  There are three possibilities:

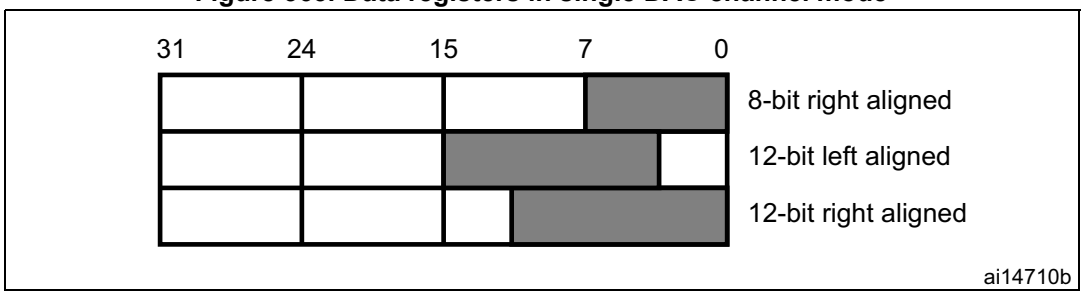
  – 8-bit right alignment: the software has to load data into the DAC_DHR8Rx[7:0] bits (stored into the DHRx[11:4] bits)

  – 12-bit left alignment: the software has to load data into the DAC_DHR12Lx[15:4] bits (stored into the DHRx[11:0] bits)

  – 12-bit right alignment: the software has to load data into the DAC_DHR12Rx[11:0] bits (stored into the DHRx[11:0] bits)

Depending on the loaded DAC_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DAC_DHRx (data holding registerx, which are internal non-memory-mapped registers). The DAC_DHRx register is then loaded into the DAC_DORx register either automatically, by software trigger or by an external event trigger.

Figure 363. Data registers in single DAC channel mode



- Dual DAC channels (when available)

  There are three possibilities:

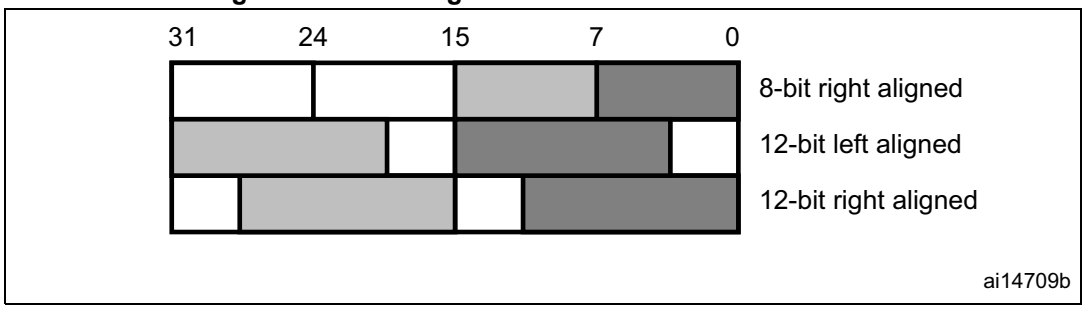
  – 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)

  – 12-bit left alignment: data for DAC channel1 to be loaded into the DAC_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)

  – 12-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12RD [27:16] bits (stored into the DHR2[11:0] bits)

Depending on the loaded DAC_DHRyyyD register, the data written by the user is shifted and stored into DHR1 and DHR2 (data holding registers, which are internal non-memory-mapped registers). The DHR1 and DHR2 registers are then loaded into the DAC_DOR1 and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

Figure 364. Data registers in dual DAC channel mode



Signed/unsigned data

DAC input data are unsigned: 0x000 corresponds to the minimum value and 0xFFF to the maximum value for 12-bit mode.

The DAC can also handle signed input data in 2's complement format. This is done by setting SINFORMATx bit in the DAC_MCR register.

When SINFORMATx bit is set, the MSB of the data written to DAC_DHRx registers is inverted when it is copied to the DAC_DORx register, and the DAC interface can accept signed data (Q1.15, Q1.11 or Q1.7 format). DAC_DHR12Lx register can be used to store 16-bit signed data in the data holding registers. The 12 MSBs of 16-bit data are used for the DAC output data and the MSB is inverted. The four LSBs are simply ignored.

Table 266. Data format (case of 12-bit data)

SINFORMATx bit	DATA written to DAC_DHRx register	DATA transfered to DAC_DORx register
0	0x000	0x000
0	0xFFF	0xFFF
1	0x7FF	0xFFF
1	0x000	0x800
1	0xFFF	0x7FF
1	0x800	0x000

27.4.5 DAC conversion

The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write operation to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12RD or DAC_DHR12LD).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one dac_ker_ck clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three dac_ker_ck clock cycles after the trigger signal.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

HFSEL bits of DAC_MCR must be set when dac_ker_ck clock speed is faster than 80 MHz.

Refer to Table *HFSEL description* below for the limitation of the DAC_DORx update rate depending on HFSEL bits and clock frequency.

If the data is updated or a software/hardware trigger event occurs during the non-allowed period, the peripheral behavior is unpredictable.

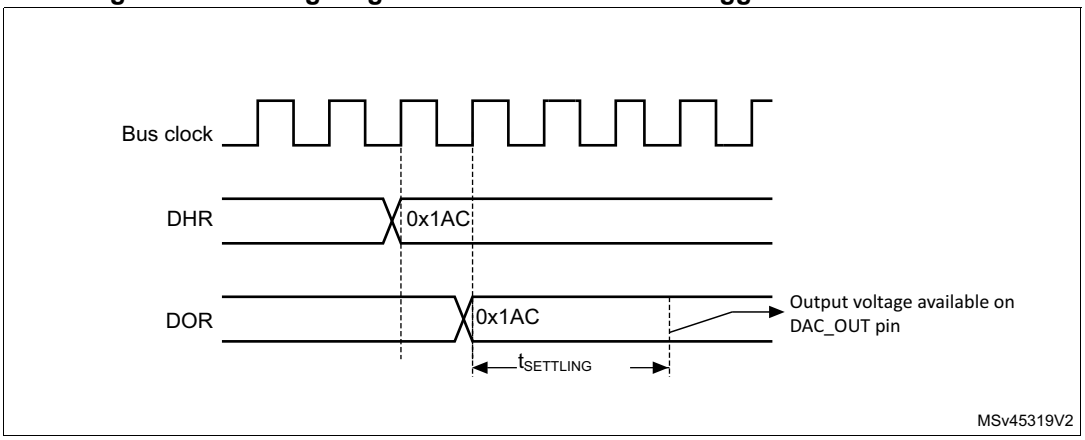
The above timing is only related to the limitation of the DAC interface. Refer also to the $t_{SETTLING}$ parameter value in the product datasheet.

Table 267. HFSEL description

HFSEL[2:0]	dac_ker_ck frequency	Latency using dac_ker_ck clock	Function
000	< 80 MHz	4	DAC_DOR update rate up to 4 dac_ker_ck cycles.
001	≥ 80 MHz(1)	5	DAC_DOR update rate up to 5 dac_ker_ck cycles.
010	≥ 160 MHz	6	DAC_DOR update rate up to 6 dac_ker_ck cycles.

1. Refer to the device datasheet for the value of the maximum kernel clock frequency.

Figure 365. Timing diagram for conversion with trigger disabled TEN = 0



27.4.6 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between $V_{REFDAC-}$ and $V_{REFDAC+}$.

The analog output voltage on the DAC channel pin is determined by the following equation:

$$\text{DAC output} = V_{REFDAC} \times \frac{DOR}{4096}$$

where all voltages are expressed in Volt.

$$V_{REFDAC} = V_{REFDAC+} - V_{REFDAC-}$$

27.4.7 DAC trigger selection

If the TENx control bit is set, the conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[4:0] control bits determine which out of 32 possible events triggers the conversion as shown in TSELx[4:0] bits of the DAC_CR register. These events can be either the software trigger or hardware triggers.

Each time a DAC interface detects a rising edge on the selected trigger source (refer to the table below), the last data stored into the DAC_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three dac_ker_ck cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

The reset trigger selection and the increment trigger selection of the sawtooth generation are performed through STRSTTRIGSELx and STINCTRIGSELx control bits, respectively. STRSTTRIGSELx mapping is similar to TSELx. Refer to the *Section 27.4.2: DAC pins and internal signals* for TSELx, STRSTTRIGSELx, and STINCTRIGSELx mappings.

Note: TSELx[4:0] bit cannot be changed when the ENx bit is set.

When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one dac_ker_ck clock cycle.

27.4.8 DMA requests

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

When an external trigger (but not a software trigger) occurs while the DMAENx bit is set, the value of the DAC_DHRx register is transferred into the DAC_DORx register when the transfer is complete, and a DMA request is generated.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, only the corresponding DMAENx bit must be set. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

As DAC_DHRx to DAC_DORx data transfer occurred before the DMA request, the very first data has to be written to the DAC_DHRx before the first trigger event occurs.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. The DAC channelx continues to convert old data.

The software must clear the DMAUDRx flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software must modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion can be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channelx, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC_CR register is enabled.

DMA double data mode

When the DMA controller is used in normal mode, only 12-bit (or 8-bit) data are transferred by a DMA request. As the AHB width is 32 bits, two 12-bit data may be transferred simultaneously. To use this mode, set the DMADOUBLEx bit of DAC_MCR register.

A DAC DMA request is generated every two external triggers (except for software triggers) when the DMAENx bit is set:

1. When the first trigger is detected, the value of the DAC_DHRx and DAC_DHRBx registers are transferred into the DAC_DORx and DAC_DORBx registers. The actual DAC data is loaded into the DAC_DORx register. A DMA request is then generated. The DMA writes the new data to the DAC_DHRx and DAC_DHRBx data registers.
2. When the next trigger is detected, the actual DAC data is loaded into the DAC_DHRBx register. This second trigger does not generate any DMA request. The DORSTATx bit indicates which DOR data is actually loaded into the analog DAC input.

DMA underrun function is also supported in DMA double data mode.

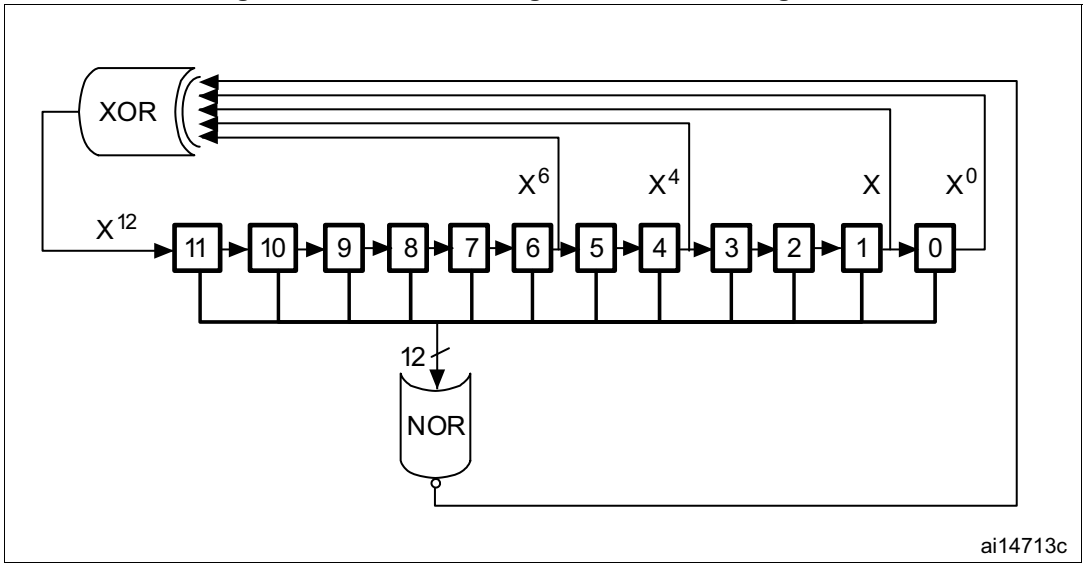
The following conditions must be met to change from double data to single data mode or vice versa:

- The DAC must be disabled.
- DMAEN bit must be cleared (ENx = 0 and DMAEN = 0).

27.4.9 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to 01. The preloaded value in LFSR is 0xAAA. This register is updated three dac_ker_ck clock cycles after each trigger event, following a specific calculation algorithm.

Figure 366. DAC LFSR register calculation algorithm

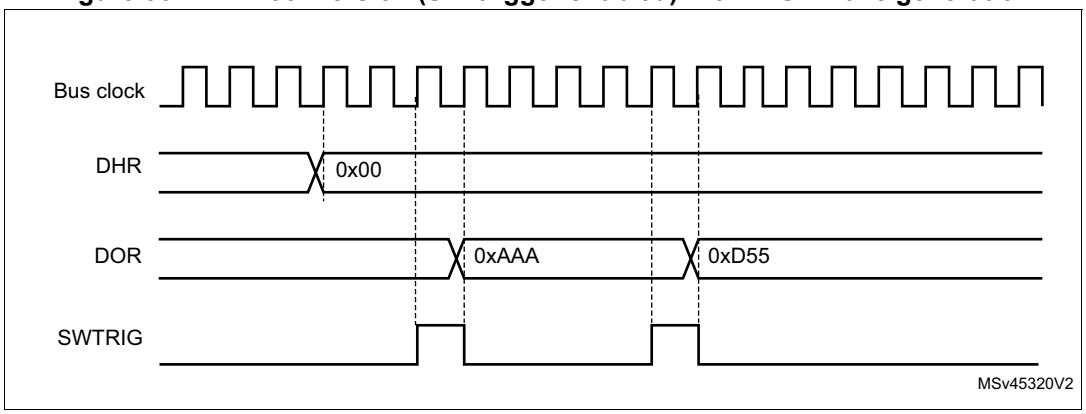


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then transferred into the DAC_DORx register.

If LFSR is 0x0000, a 1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Figure 367. DAC conversion (SW trigger enabled) with LFSR wave generation



Note: The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.

27.4.10 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to 10. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three dac_ker_ck clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is transferred into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

Figure 368. DAC triangle wave generation

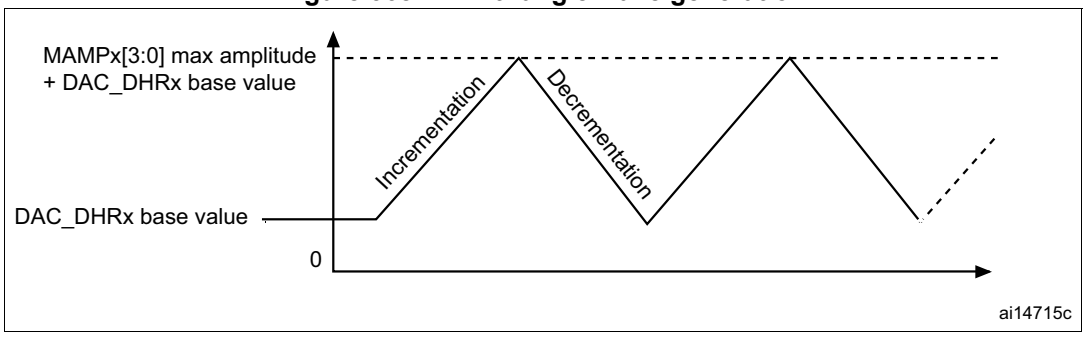
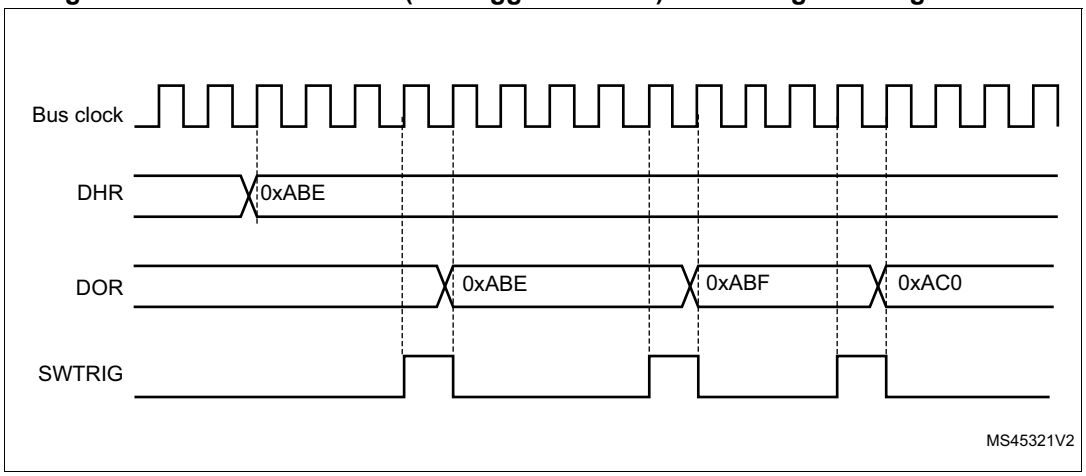


Figure 369. DAC conversion (SW trigger enabled) with triangle wave generation



Note: The DAC trigger must be enabled for triangle wave generation by setting the TENx bit in the DAC_CR register.

The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

27.4.11 DAC sawtooth wave generation

The DAC can generate a sawtooth waveform. Specific register settings for the initial value, increment value and direction control are required:

- DAC sawtooth wave generation is selected by setting WAVEx[1:0] to 11 in the DAC_CR register.
- The sawtooth counter initial value (reset value) is configured through STRSTDATAx[11:0] bits in the DAC_STRx register.
- The increment value is defined by the STINCDATAx[15:0] bits in the DAC_STRx register.
- The sawtooth direction is defined by STDIRx bit in the DAC_STRx register.

The sawtooth counter starts from STRSTDATAx[11:0] (bits 12 to 15 are set to 0b0000), each increment trigger then increments (or decrements) STINCDATAx[15:0] value.

The DAC output is used from 12 MSBs of those counter value. When the counter reaches 0x0000 or 0xFFFF, the value is saturated. The sawtooth reset trigger signal initializes the counter value to the STRSTDATAx[11:0] (bits 12 to 15 are set to 0b0000) value.

The increment trigger (STINCTRIG) and reset trigger (STRSTTRIG) must be selected through the STINCTRIGSELx[3:0] and the STRSTTRIGSELx[3:0] bits.

Figure 370. DAC sawtooth wave generation (STDIRx = 0)

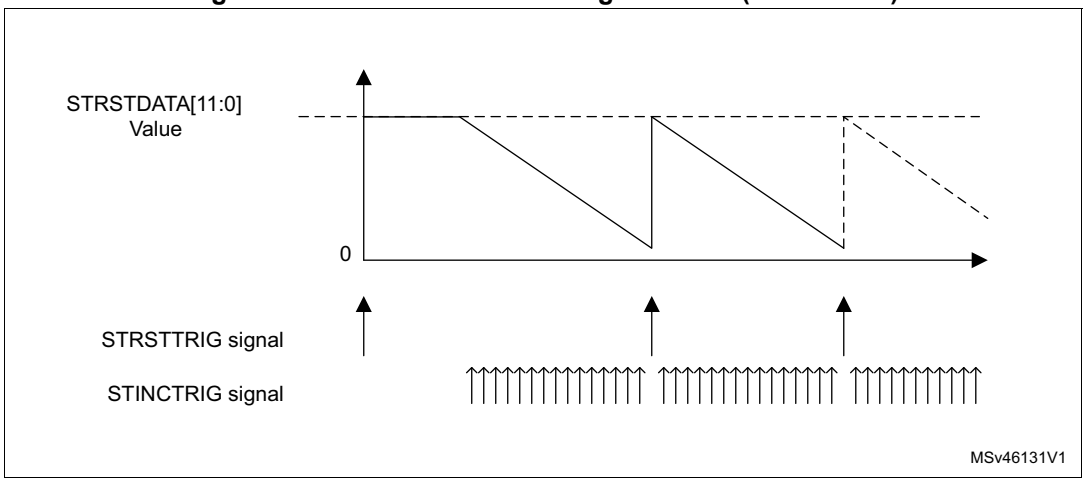
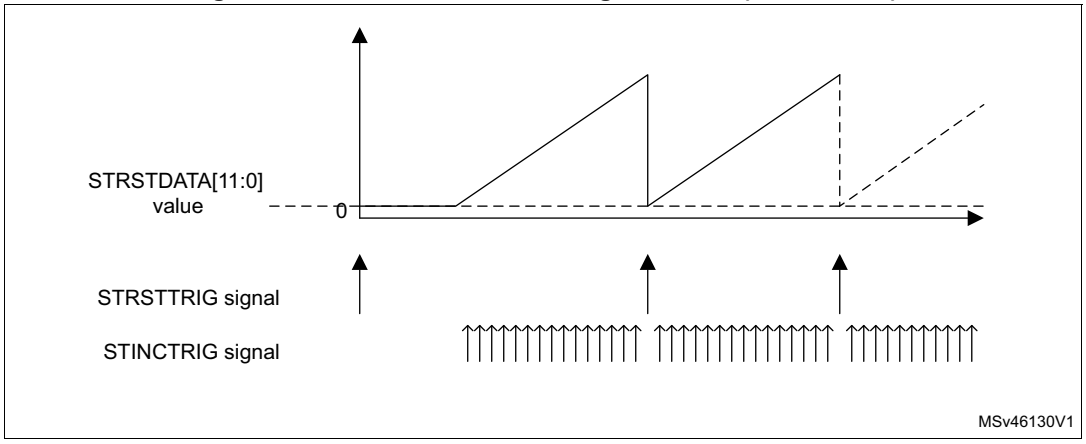
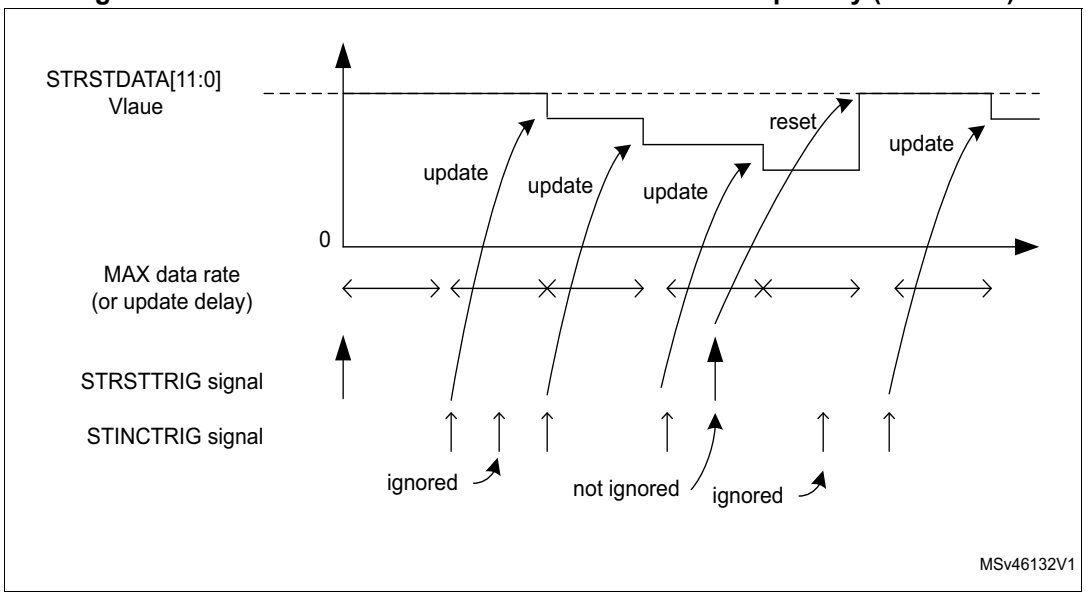


Figure 371. DAC sawtooth wave generation (STDIRx = 1)



The STRSTTRIG signal has higher priority than STINCTRIG. The trigger signal cannot be faster than the DAC_DORx update rate (see *Table HFSEL description - if available - for details)*. If STINCTRIG is asserted faster than the allowed data update rate, the STINCTRIG trigger is ignored. If the STRSTTRIG signal is applied after the STINCTRIG and before DAC_DORx update rate constraints, STRSTTRIG is put on hold. Then, immediately after the data increment, the reset trigger is applied.

Figure 372. DAC sawtooth STINCTRIG and STRSTTRIG priority (STDIR = 0)



27.4.12 DAC extended sawtooth wave generation

The extended sawtooth wave generation is more flexible than the standard sawtooth wave generation described in *Section 27.4.11: DAC sawtooth wave generation*:

- It is possible to define the minimum and maximum output values.
- The extended sawtooth is managed by a signed 20-bit internal counter, enabling to convert data in 16.4 format.

  The first MSB is the sign bit, the second to the forth MSBs contain the overflow part, the middle 12 bits (5 to 16 MSBs) are used for the DAC output data, and the four LSBs consist of the fractional part. When the four MSBs equal 0b0000, the counter value is transferred to the DAC output. Otherwise, the DAC output value is saturated at 0x000, 0xFFF, or at the minimum or maximum value, programmed in the DAC_STEXTMMx register.

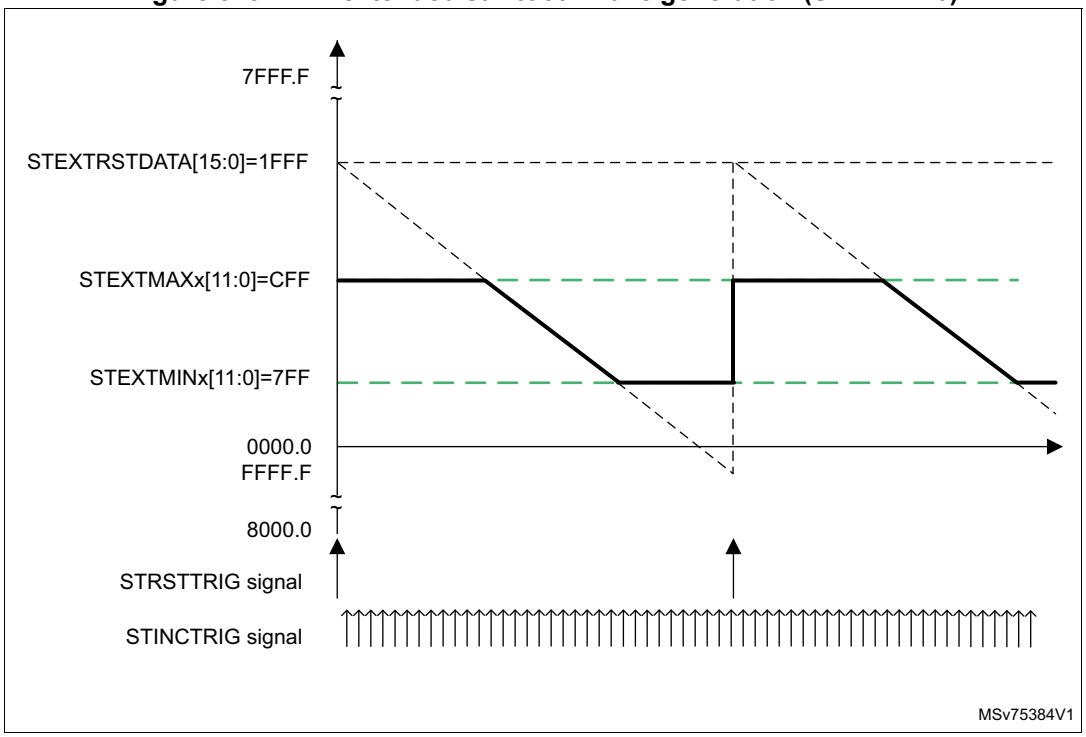
The following sequence is required to configure and start the extended sawtooth generation:

1. Clear the ENx bit of the DAC_CR register.
2. Activated the extended sawtooth mode by setting the STEXTx bit of the DAC_STRx register.
3. Configure the sawtooth counter initial value (reset value) through the STEXTRSTDATAx[15:0] bitfield of the DAC_STEXTRST register.
4. Configure the increment value through the STINCDATAx[15:0] bitfield of the DAC_STRx register whose four LSBs correspond to the fractional part.
5. Configure the sawtooth minimum and maximum values through the STEXTMINx[11:0] and STEXTMAXx[11:0] bitfield of the DAC_STEXTMMx register
6. Configure the sawtooth direction through the STDIR1 bit of the DAC_STR1 register.
7. Select the sawtooth reset and increment trigger through the STRSTTRIGSELx[3:0] and STINCTRIGSELx[3:0] bitfields.
8. Set WAVEx[1:0] to 0b11 and ENx to 1.
9. Generate a sawtooth reset trigger to start the sawtooth generation.

When STDIR1 is cleared (decrement case), the internal counter operates as follows:

- When the STRSTTRIG is applied, the counter 16 MSBs are reset to the signed 20-bit STEXTRSTDATAx[15:0] value, and the four LSBs are reset to 0b0000.
- When the STINCTRIG is applied, the counter is subtracted from STINCDATAx[15:0].
- When the counter four MSBs are greater than or equal to 0b0001, the DAC output is saturated at the STEXTMAXx[11:0] value.
- When the counter four MSBs are equal to 0b0000, the counter middle 12-bit value is evaluated. If it is lower than STEXTMAXx[11:0] and greater than STEXTMINx[11:0], the value is transferred to the DAC output.
- When the counter middle 12-bit value is lower than STEXTMINx[11:0], the DAC output value is saturated at STEXTMINx[11:0].
- When the counter value reaches 0x0000.0, the counter continues down-counting below the negative value 0xFFFFF. If the value reaches 0x8000.0, the internal counter value is saturated at 0x8000.0.

Figure 373. DAC extended sawtooth wave generation (STDIRx = 0)

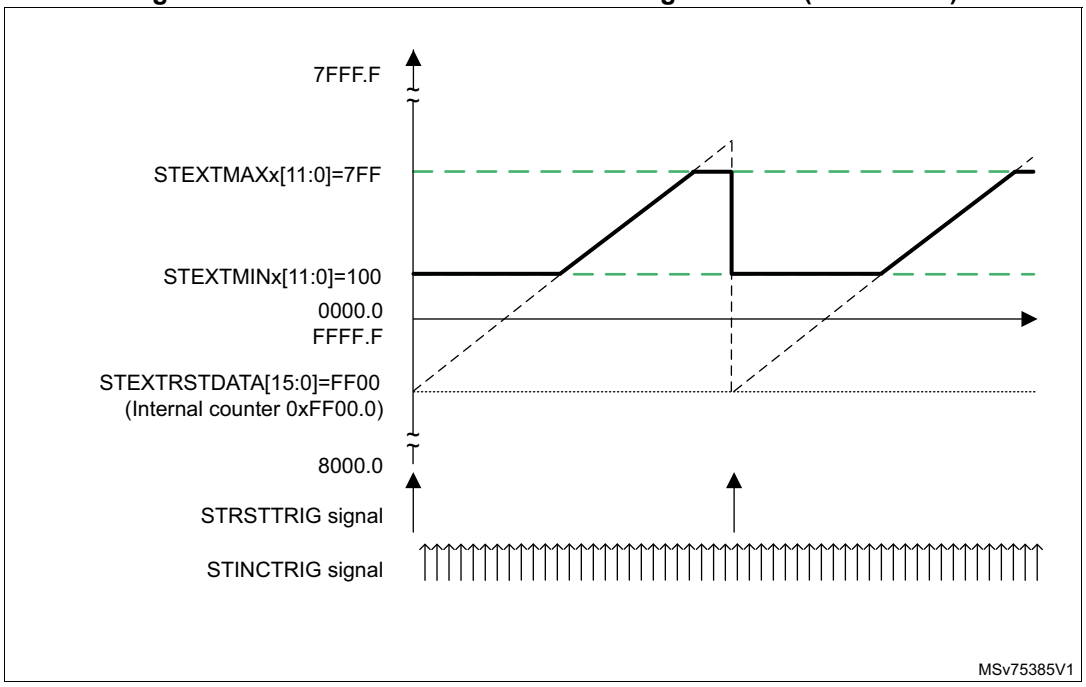


1. The line in bold represents the DAC output, the dashed line in black the internal counter value, and the dashed lines in green the extended sawtooth minimum and maximum values.

When STDIRx is set (increment case), the internal counter operates as follows:

- When the STRSTTRIG is applied, the counter 16 MSBs are reset to STEXTRSTDATAx[15:0] value, and the four LSBs are reset to 0x0.
- When the STINCTRIG is applied, the counter is added to STINCDATA1[15:0].
- When the counter is negative (MSB = 1), the DAC output is saturated at the STEXTMINx[11:0] value.
- When the counter four MSBs are equal to 0b0000, the counter middle 12-bit value is evaluated. If it is lower than STEXTMAXx[11:0] and greater than STEXTMINx[11:0], the value is transferred to the DAC output.
- When the counter middle 12-bit value is greater than the STEXTMAXx[11:0] value, the DAC output value is saturated at STEXTMAXx[11:0].
- When the value reaches 0x7FFF.F, the counter is saturated.

Figure 374. DAC extended sawtooth wave generation (STDIRx = 1)



1. The line in bold represents the DAC output, the dashed line in black the internal counter value, and the dashed lines in green the extended sawtooth minimum and maximum values.

Dynamic modification of the extended sawtooth parameters

The STEXTRSTDATAx[15:0] bitfield can be modified while an extended sawtooth generation is ongoing. The new value is taken into account when a new STRSTTRIG is triggered.

When STINCDATAx[15:0] and/or STDIRx are modified, the new value is applied when a new STINCTRIG is triggered.

When either or both of STEXTMAXx[11:0] and STEXTMINx[11:0] are modified, the new value is applied when the next STRSTTRIG or STINCTRIG is triggered. To ensure STEXTMAXx[11:0] is always greater than STEXTMINx[11:0], it is recommended to program them simultaneously through the DAC_STEXTMMx register.

Since all the extended sawtooth parameters are stored in registers located in the AHB domain, and the DAC operates from the kernel clock, if an STRSTTRIG or an STINCTRIG

signal arrives while a sawtooth parameter is being updated, the old value may be taken into account instead of the new one. To prevent this issue from occurring, the extended sawtooth parameters must be written much before the next trigger.

27.4.13 DAC channel modes

Each DAC channel can be configured in normal mode or sample and hold mode. The output buffer can be enabled to obtain a high drive capability. Before enabling output buffer, the voltage offset needs to be calibrated. This calibration is performed at the factory (loaded after reset) and can be adjusted by software during application operation.

Normal mode

In normal mode, there are four combinations, by changing the buffer state and by changing the DACx_OUTy pin interconnections.

To enable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:

- 000: DAC is connected to the external pin
- 001: DAC is connected to external pin and to on-chip peripherals

To disable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:

- 010: DAC is connected to the external pin
- 011: DAC is connected to on-chip peripherals

Sample and hold mode

In sample and hold mode, the DAC core converts data on a triggered conversion, and then holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A stabilization period, which value depends on the buffer state, is required before each new conversion.

In this mode, the DAC core and all corresponding logic and registers are driven by the LSI/LSE low-speed clock (dac_hold_ck) in addition to the dac_ker_ck clock, allowing using the DAC channels in deep low power modes such as Stop mode.

The LSI/LSE low-speed clock (dac_hold_ck) must not be stopped when the sample and hold mode is enabled.

The sample/hold mode operations can be divided into three phases:

1. Sample phase: the sample/hold element is charged to the desired voltage. The charging time depends on capacitor value (internal or external, selected by the user). The sampling time is configured with the TSAMPLEx[9:0] bits in DAC_SHSRx register. During the write of the TSAMPLEx[9:0] bits, the BWSTx bit in DAC_SR register is set to 1 to synchronize between both clocks domains (AHB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation
2. Hold phase: the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the THOLDx[9:0] bits in DAC_SHHR register
3. Refresh phase: the refresh time is configured with the TREFRESHx[7:0] bits in DAC_SHRR register

The timings for the three phases above are in units of LSI/LSE clock periods. As an example, to configure a sample time of 350 µs, a hold time of 2 ms and a refresh time of 100 µs assuming LSI/LSE ~32 KHz is selected:

- 12 cycles are required for sample phase: TSAMPLEx[9:0] = 11.
- 62 cycles are required for hold phase: THOLDx[9:0] = 62.
- and 4 cycles are required for refresh period: TREFRESHx[7:0] = 4.

In this example, the power consumption is reduced by almost a factor of 15 versus normal modes.

The formulas to compute the right sample and refresh timings are described in the table below, the Hold time depends on the leakage current.

Table 268. Sample and refresh timings

Buffer State	t_{SAMP} (1)(2)	$t_{REFRESH}$ (2)(3)
Enable	7 µs + (10*R_{BON}*C_{SH})	7 µs + (R_{BON}*C_{SH})*ln(2*N_{LSB})
Disable	3 µs + (10*R_{BOFF}*C_{SH})	3 µs + (R_{BOFF}*C_{SH})*ln(2*N_{LSB})

1. In the above formula, the settling to the desired code value with ½ LSB or accuracy requires 10 constant time for 12 bits resolution. For 8-bit resolution, the settling time is 7 constant time.
2. C_{SH} is the capacitor in sample and hold mode.
3. The tolerated voltage drop during the hold phase “Vd” is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with ½ LSB error accuracy requires ln(2*Nlsb) constant time of the DAC.

Example of the sample and refresh time calculation with output buffer on

The values used in the example below are provided as indication only (refer to the datasheet for product data):

C_{SH} = 100 nF

$V_{REFDAC+}$ = 3.0 V

Sampling phase:

t_{SAMP} = 7 µs + (10 * 2000 * 100 * 10^{-9}) = 2.007 ms

(where R_{BON} = 2 kΩ)

Refresh phase:

$t_{REFRESH}$ = 7 µs + (2000 * 100 * 10^{-9}) * ln(2*10) = 606.1 µs

(where N_{LSB} = 10 (10 LSB drop during the hold phase)

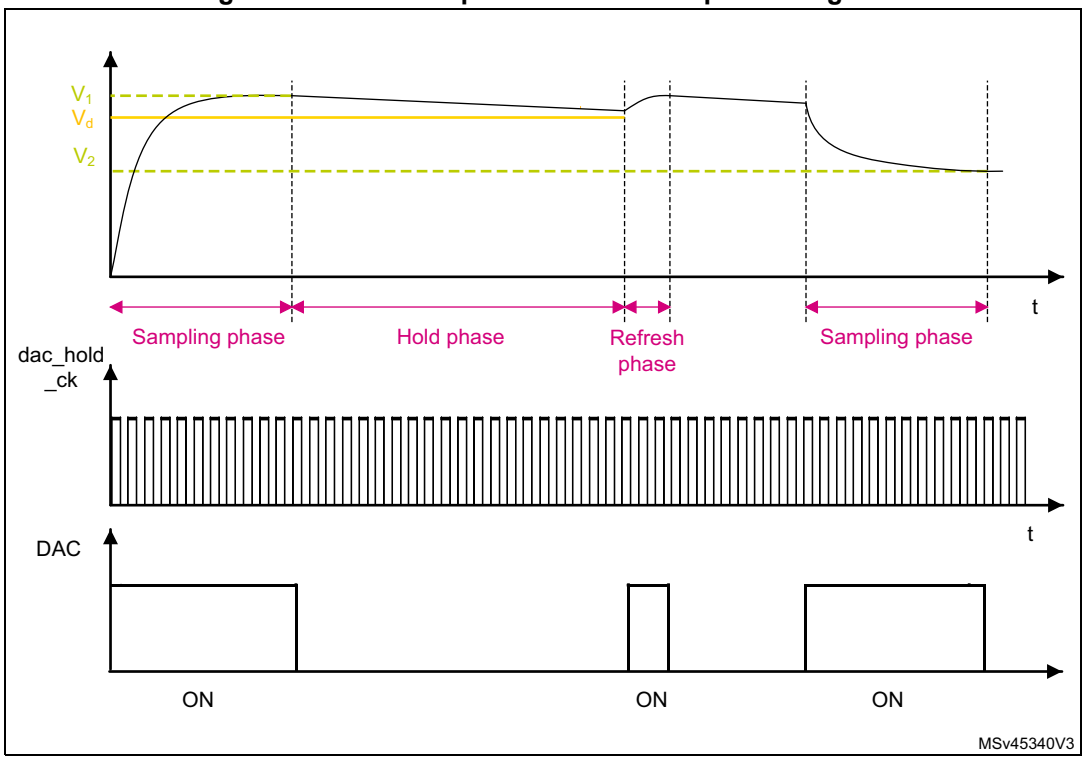
Hold phase:

D_v = i_{leak} * t_{hold} / C_{SH} = 0.0073 V (10 LSB of 12bit at 3 V)

i_{leak} = 150 nA (worst case on the IO leakage on all the temperature range)

t_{hold} = 0.0073 * 100 * 10^{-9} / (150 * 10^{-9}) = 4.867 ms

Figure 375. DAC sample and hold mode phase diagram



Like in normal mode, the sample and hold mode has different configurations.

To enable the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:

- 100: DAC is connected to the external pin
- 101: DAC is connected to external pin and to on chip peripherals

To disabled the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:

- 110: DAC is connected to external pin and to on chip peripherals
- 111: DAC is connected to on chip peripherals

  When MODEx[2:0] bits are equal to 111, an internal capacitor, C_{Lint}, holds the voltage output of the DAC core and then drive it to on-chip peripherals.

All sample and hold phases are interruptible, and any change in DAC_DHRx immediately triggers a new sample phase.

Note: The sawtooth wave generation is not supported in sample and hold mode operation.

Table 269. Channel output modes summary

MODEx[2:0]			Mode	Buffer	Output connections
0	0	0	Normal mode	Enabled	Connected to external pin
0	0	1			Connected to external pin and to on chip-peripherals (such as comparators)
0	1	0		Disabled	Connected to external pin
0	1	1			Connected to on chip peripherals (such as comparators)

Table 269. Channel output modes summary (continued)

MODEx[2:0]			Mode	Buffer	Output connections
1	0	0	Sample and hold mode	Enabled	Connected to external pin
1	0	1			Connected to external pin and to on chip peripherals (such as comparators)
1	1	0		Disabled	Connected to external pin and to on chip peripherals (such as comparators)
1	1	1			Connected to on chip peripherals (such as comparators)

27.4.14 DAC channel buffer calibration

The transfer function for an N-bit digital-to-analog converter (DAC) is:

$$V_{out} = ((D / 2^N) \times G \times V_{REFDAC}) + V_{OS}$$

Where V_{OUT} is the analog output, D is the digital input, G is the gain, V_{REFDAC} is the nominal full-scale voltage, and V_{OS} is the offset voltage. For an ideal DAC channel, G = 1 and V_{OS} = 0.

Due to output buffer characteristics, the voltage offset may differ from part-to-part and introduce an absolute offset error on the analog output. To compensate the V_{OS}, a calibration is required by a trimming technique.

The calibration is only valid when the DAC channelx is operating with buffer enabled (MODEx[2:0] = 0b000 or 0b001 or 0b100 or 0b101). if applied in other modes when the buffer is off, it has no effect. During the calibration:

- The buffer output is disconnected from the pin internal/external connections and put in tristate mode (HiZ).
- The buffer acts as a comparator to sense the middle-code value 0x800 and compare it to V_{REFDAC}/2 signal through an internal bridge, then toggle its output signal to 0 or 1 depending on the comparison result (CAL_FLAGx bit).

Two calibration techniques are provided:

- Factory trimming (default setting)

  The DAC buffer offset is factory trimmed. The default value of OTRIMx[4:0] bits in DAC_CCR register is the factory trimming value and it is loaded once DAC digital interface is reset.
- User trimming

  The user trimming can be done when the operating conditions differs from nominal factory trimming conditions and in particular when V_{DDA} voltage, temperature, V_{REFDAC} values change and can be done at any point during application by software.

Note: Refer to the datasheet for more details of the nominal factory trimming conditions.

In addition, when V_{DD} is removed (example the device enters in Standby or VBAT modes) the calibration is required.

The steps to perform a user trimming calibration are as below:

1. If the DAC channel is active, write 0 to ENx bit in DAC_CR to disable the channel.
2. Select a mode where the buffer is enabled, by writing to DAC_MCR register, MODEx[2:0] = 0b000 or 0b001 or 0b100 or 0b101.
3. Start the DAC channelx calibration, by setting the CENx bit in DAC_CR register to 1.
4. Apply a trimming algorithm:
   a) Write a code into OTRIMx[4:0] bits, starting by 0b00000.
   b) Wait for t_{TRIM} delay.
   c) Check if CAL_FLAGx bit in DAC_SR is set to 1.
   d) Until the CAL_FLAGx is read as 1 or the maximum trimming code is reached, increment OTRIMx[4:0] and repeat substeps from (b) to (d).

The software algorithm may use either a successive approximation or dichotomy techniques to compute and set the content of OTRIMx[4:0] bits in a faster way.

Note: A t_{TRIM} delay must be respected between the write to the OTRIMx[4:0] bits and the read of the CAL_FLAGx bit in DAC_SR register in order to get a correct value.This parameter is specified into datasheet electrical characteristics section.

If V_{DDA}, V_{REFDAC} and temperature conditions do not change during device operation while it enters more often in Standby and VBAT modes, the software may store the OTRIMx[4:0] bits found in the first user calibration in the flash or in back-up registers. then to load/write them directly when the device power is back again thus avoiding to wait for a new calibration time.

When CENx bit is set, it is not allowed to set ENx bit.

27.4.15 DAC channel conversion modes

Four conversion modes are possible.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TENx.
2. Configure the trigger sources by setting different values in the TSELx[4:0] bits.
3. Load the DAC channel data into the desired DHR registers (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the DHRx register is transferred into DAC_DORx (three dac_ker_ck clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TENx.
2. Configure the trigger sources by setting different values in the TSELx[4:0] bits.
3. Configure the DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the DAC channel data into the desired DHR register (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_ker_ck clock cycles later). Then the LFSR1 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bits, TENx.
2. Configure the trigger sources by setting different values in the TSELx[4:0] bits.
3. Configure the DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the DAC channel data into the desired DHR register (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the DAC channel triangle counter, with the same triangle amplitude, is added to the DHRx register and the sum is transferred into DAC_DOR1 (three dac_ker_ck clock cycles later). The DAC channel triangle counter is then updated.

Independent trigger with single sawtooth generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure the trigger sources by setting different values in STRSTTRIGSELx[3:0] and STINCTRIGSELx[3:0] bits.
2. Configure the DAC channel WAVEx[1:0] bits to 11 and set the same STRSTDATAx[11:0], STINCDATAx[15:0] and STDIRx values for each register.

When a DAC channel trigger arrives, the DAC channel sawtooth counter updates the DHRx register and transfers it into DAC_DOR1 (three AHB clock cycles later).

27.4.16 Dual DAC channel conversion modes (if dual channels are available)

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time. For the wave generation, no accesses to DHRxxxD registers are required. As a result, two output channels can be used either independently or simultaneously.

15 conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DAC_DHRx registers if needed.

All modes are described in the paragraphs below.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three dac_ker_ck clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three dac_ker_ck clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_ker_ck clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_ker_ck clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_ker_ck clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_ker_ck clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_ker_ck clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_ker_ck clock cycles later). The DAC channel2 triangle counter is then updated.

Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_ker_ck clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_ker_ck clock cycles later). The DAC channel2 triangle counter is then updated.

Independent trigger with single sawtooth generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure different trigger sources by setting different values in STRSTTRIGSEL1[3:0], STRSTTRIGSEL2[3:0], STINCTRIGSEL2[3:0] and STINCTRIGSEL1[3:0] bits.
2. Configure the two DAC channel WAVEx[1:0] bits to 11 and set the same STRSTDATAx[11:0], STINCDATAx[15:0] and STDIRx values for each register.

When a DAC channel1 trigger arrives, the DAC channel1 sawtooth counter updates the DHR1 register and transfers it into DAC_DOR1 (three AHB clock cycles later).

When a DAC channel2 trigger arrives, the DAC channel2 sawtooth counter updates the DHR1 register and transfers it into DAC_DOR1 (three AHB clock cycles later).

Independent trigger with different sawtooth wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure different trigger sources by setting different values in the STRSTTRIGSEL1[23:0], STRSTTRIGSEL2[23:0], STINCTRIGSEL2[3:0] and STINCTRIGSEL1[3:0] bits.
2. Configure the two DAC channel WAVEx[1:0] bits as 11 and set different STRSTDATAx[11:0], STINCDATAx[15:0] and STDIRx value for each register.

When a DAC channel1 trigger arrives, the DAC channel1 sawtooth counter updates the DHR1 register and loads it into DAC_DOR1 (three AHB clock cycles later).

When a DAC channel2 trigger arrives, the DAC channel2 sawtooth counter updates the DHR2 register and loads it into DAC_DOR1 (three AHB clock cycles later).

Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

In this configuration, one dac_ker_ck clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three dac_ker_ck clock cycles).

Simultaneous trigger with single LFSR generation

1. To configure the DAC in this conversion mode, the following sequence is required:
2. Set the two DAC channel trigger enable bits TEN1 and TEN2.
3. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
4. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
5. Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD).

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_ker_ck clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same

mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_ker_ck clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_ker_ck clock cycles later). The LFSR1 counter is then updated.
At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_ker_ck clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value using the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_ker_ck clock cycles later). The DAC channel1 triangle counter is then updated.

At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_ker_ck clock cycles later). The DAC channel2 triangle counter is then updated.

Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into

DAC_DOR1 (three AHB clock cycles later). Then the DAC channel1 triangle counter is updated.

At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_ker_ck clock cycles later). Then the DAC channel2 triangle counter is updated.

Simultaneous trigger with single sawtooth wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure the same trigger source for both DAC channels by setting the same value in STRSTTRIGSEL1[3:0],STRSTTRIGSEL2[3:0], STINCTRIGSEL2[3:0] and STINCTRIGSEL1[3:0] bits.
2. Configure the two DAC channel WAVEx[1:0] bits to 11 and set the same STRSTDATAx[11:0], STINCDATAx[15:0] and STDIRx value for each register.

When a trigger arrives, the DAC channel1/2 sawtooth counter updates DHR1 and DHR2 registers and loads them into DAC_DOR1/2 (three AHB clock cycles later).

Simultaneous trigger with different sawtooth wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure the same trigger source for both DAC channels by setting the same value in STRSTTRIGSEL1[3:0], STRSTTRIGSEL2[3:0], STINCTRIGSEL2[3:0] bits and STINCTRIGSEL1[3:0] bits.
2. Configure the two DAC channel WAVEx[1:0] bits to 11 and set different STRSTDATAx[11:0], STINCDATAx[15:0], STDIRx values for each register.

When a trigger arrives, DAC channel1/2 sawtooth counter updates the DHR1 and DHR2 registers and loads them independently into DAC_DOR1/2 (three AHB clock cycles later).

27.5 DAC in low-power modes

Table 270. Effect of low-power modes on DAC

Mode	Description
Sleep	No effect, DAC used with DMA.
Stop(1)	The DAC remains active with a static value. The sample and hold mode can be selected using LSE/LSI clock.
Standby	The DAC peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 27.3: DAC implementation* for information on the Stop modes supported by the DAC peripheral.

27.6 DAC interrupts

Table 271. DAC interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby mode
DAC	DMA underrun	DMAUDRx	DMAUDRIEx	Write DMAUDRx = 1	Yes	No	No

27.7 DAC registers

Refer to *Section 1* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

27.7.1 DAC control register (DAC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSEL2[4]	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[3:0]				TEN2	EN2
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSEL1[4]	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3:0]				TEN1	EN1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 30 **CEN2**: DAC channel2 calibration enable

This bit is set and cleared by software to enable/disable DAC channel2 calibration, it can be written only if EN2 bit is set to 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel2 in normal operating mode

1: DAC channel2 in calibration mode

Note: This bit is available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bit 29 **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable

This bit is set and cleared by software.

0: DAC channel2 DMA underrun interrupt disabled

1: DAC channel2 DMA underrun interrupt enabled

Note: This bit is available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bit 28 **DMAEN2**: DAC channel2 DMA enable

This bit is set and cleared by software.

0: DAC channel2 DMA mode disabled

1: DAC channel2 DMA mode enabled

Note: This bit is available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bits 27:24 **MAMP2[3:0]**: DAC channel2 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Note: These bits are available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bits 23:22 **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable

These bits are set/reset by software.

00: wave generation disabled
01: Noise wave generation enabled
10: Triangle wave generation enabled
11: Sawtooth wave generation enabled

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)
These bits are available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bits 31, 21:18 **TSEL2[4:0]**: DAC channel2 trigger selection

These bits select the external event used to trigger DAC channel2

00000: SWTRIG2
00001: dac_ch2_trg1
00010: dac_ch2_trg2
...
11111: dac_ch2_trg31

For details on trigger configuration and mapping, refer to *Section 27.4.2: DAC pins and internal signals*.

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).
These bits are available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bit 17 **TEN2**: DAC channel2 trigger enable

This bit is set and cleared by software to enable/disable DAC channel2 trigger

0: DAC channel2 trigger disabled and data written into the DAC_DHR2 register are transferred one dac_ker_ck clock cycle later to the DAC_DOR2 register
1: DAC channel2 trigger enabled and data from the DAC_DHR2 register are transferred three dac_ker_ck clock cycles later to the DAC_DOR2 register

Note: When software trigger is selected, the transfer from the DAC_DHR2 register to the DAC_DOR2 register takes only one dac_ker_ck clock cycle.
These bits are available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bit 16 **EN2**: DAC channel2 enable

This bit is set and cleared by software to enable/disable DAC channel2.

0: DAC channel2 disabled

1: DAC channel2 enabled

Note: These bits are available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bit 14 **CEN1**: DAC channel1 calibration enable

This bit is set and cleared by software to enable/disable DAC channel1 calibration, it can be written only if bit EN1 = 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel1 in normal operating mode

1: DAC channel1 in calibration mode

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.

0: DAC channel1 DMA Underrun Interrupt disabled

1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.

0: DAC channel1 DMA mode disabled

1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1

0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3

0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7

0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15

0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31

0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63

0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127

0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255

1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511

1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023

1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047

≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

These bits are set and cleared by software.

00: wave generation disabled

01: Noise wave generation enabled

10: Triangle wave generation enabled

11: Sawtooth wave generation enabled

Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 15, 5:2 **TSEL1[4:0]**: DAC channel1 trigger selection
These bits select the external event used to trigger DAC channel1
00000: SWTRIG1
00001: dac_ch1_trg1
00010: dac_ch1_trg2
...
11111: dac_ch1_trg31
For details on trigger configuration and mapping, refer to *Section 27.4.2: DAC pins and internal signals*.

Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 1 **TEN1**: DAC channel1 trigger enable
This bit is set and cleared by software to enable/disable DAC channel1 trigger.
0: DAC channel1 trigger disabled and data written into the DAC_DHR1 register are transferred one dac_ker_ck clock cycle later to the DAC_DOR1 register
1: DAC channel1 trigger enabled and data from the DAC_DHR1 register are transferred three dac_ker_ck clock cycles later to the DAC_DOR1 register

Note: When software trigger is selected, the transfer from the DAC_DHR1 *register to the DAC_DOR1 register takes only one dac_ker_ck clock cycle.*

Bit 0 **EN1**: DAC channel1 enable
This bit is set and cleared by software to enable/disable DAC channel1.
0: DAC channel1 disabled
1: DAC channel1 enabled

27.7.2 DAC software trigger register (DAC_SWTRGR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIGB2	SWTRIGB1
														w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
														w	w

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

Bit 17 **SWTRIGB2**: DAC channel2 software trigger B

This bit is set by software to trigger the DAC in software trigger mode (sawtooth generation) It is cleared by hardware.

0: No trigger

1: Trigger for sawtooth increment

Note: This bit is available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bit 16 **SWTRIGB1**: DAC channel1 software trigger B

This bit is set by software to trigger the DAC in software trigger mode (sawtooth generation) It is cleared by hardware.

0: No trigger

1: Trigger for sawtooth increment

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

Bit 1 **SWTRIG2**: DAC channel2 software trigger

This bit is set by software to trigger the DAC in software trigger mode, or software reset trigger mode for the sawtooth wave generation.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_ker_ck clock cycle later) once the DAC_DHR2 register value has been loaded into the DAC_DOR2 register.

This bit is available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set by software to trigger the DAC in software trigger mode, or software reset trigger mode for the sawtooth wave generation.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_ker_ck clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

27.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC1DHRB[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC1DHRB[11:0]**: DAC channel1 12-bit right-aligned data B
These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in double data mode.

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

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel1.

27.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC1DHRB[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC1DHRB[11:0]**: DAC channel1 12-bit left-aligned data B
These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in double data mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software.
They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

27.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHRB[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC1DHRB[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1 when the DAC operates in double data mode.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1.

27.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)

This register is available only on dual-channel DACs. Refer to *Section 27.3: DAC implementation*.

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DHRB[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DHRB[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in DMA double data mode.

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

Bits 11:0 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel2.

27.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)

This register is available only on dual-channel DACs. Refer to *Section 27.3: DAC implementation*.

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHRB[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHRB[11:0]**: DAC channel2 12-bit left-aligned data B
These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in double data mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0 Reserved, must be kept at reset value.

27.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)

This register is available only on dual-channel DACs. Refer to *Section 27.3: DAC implementation*.

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHRB[7:0]								DACC2DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC2DHRB[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel2 when the DAC operates in double data mode.

Bits 7:0 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

27.7.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

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

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

27.7.10 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

27.7.11 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel1.

27.7.12 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC1DORB[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC1DORB[11:0]**: DAC channel1 data output
These bits are read-only. They contain data output for DAC channel1 B.

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

Bits 11:0 **DACC1DOR[11:0]**: DAC channel1 data output
These bits are read-only, they contain data output for DAC channel1.

27.7.13 DAC channel2 data output register (DAC_DOR2)

This register is available only on dual-channel DACs. Refer to *Section 27.3: DAC implementation*.

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DORB[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DORB[11:0]**: DAC channel2 data output
These bits are read-only. They contain data output for DAC channel2 B.

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

Bits 11:0 **DACC2DOR[11:0]**: DAC channel2 data output
These bits are read-only, they contain data output for DAC channel2.

27.7.14 DAC status register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BWST2	CAL_FLAG2	DMAUDR2	DORSTAT2	DAC2RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1	r	r											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BWST1	CAL_FLAG1	DMAUDR1	DORSTAT1	DAC1RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1	r	r											

Bit 31 **BWST2**: DAC channel2 busy writing sample time flag

This bit is systematically set just after sample and hold mode enable. It is set each time the software writes the register DAC_SHSR2, It is cleared by hardware when the write operation of DAC_SHSR2 is complete. (It takes about 3 LSI/LSE periods of synchronization).

0:There is no write operation of DAC_SHSR2 ongoing: DAC_SHSR2 can be written

1:There is a write operation of DAC_SHSR2 ongoing: DAC_SHSR2 cannot be written

Note: This bit is available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bit 30 **CAL_FLAG2**: DAC channel2 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Note: This bit is available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bit 29 **DMAUDR2**: DAC channel2 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel2

1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate).

Note: This bit is available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bit 28 **DORSTAT2**: DAC channel2 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in double data mode.

0: DOR[11:0] is used actual DAC output

1: DORB[11:0] is used actual DAC output

Note: This bit is available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bit 27 **DAC2RDY**: DAC channel2 ready status bit

This bit is set and cleared by hardware.

0: DAC channel2 is not yet ready to accept the trigger nor output data

1: DAC channel2 is ready to accept the trigger or output data

Note: This bit is available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bits 26:16 Reserved, must be kept at reset value.

Bit 15 **BWST1**: DAC channel1 busy writing sample time flag

This bit is systematically set just after sample and hold mode enable and is set each time the software writes the register DAC_SHSR1, It is cleared by hardware when the write operation of DAC_SHSR1 is complete. (It takes about 3 LSI/LSE periods of synchronization).

0:There is no write operation of DAC_SHSR1 ongoing: DAC_SHSR1 can be written

1:There is a write operation of DAC_SHSR1 ongoing: DAC_SHSR1 cannot be written

Bit 14 **CAL_FLAG1**: DAC channel1 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel1

1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bit 12 **DORSTAT1**: DAC channel1 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in double data mode.

0: DOR[11:0] is used actual DAC output

1: DORB[11:0] is used actual DAC output

Bit 11 **DAC1RDY**: DAC channel1 ready status bit

This bit is set and cleared by hardware.

0: DAC channel1 is not yet ready to accept the trigger nor output data

1: DAC channel1 is ready to accept the trigger or output data

Bits 10:0 Reserved, must be kept at reset value.

27.7.15 DAC calibration control register (DAC_CCR)

Address offset: 0x38

Reset value: 0x00XX 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4:0]				
											rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **OTRIM2[4:0]**: DAC channel2 offset trimming value

These bits are available only on dual-channel DACs. Refer to *Section 27.3: DAC implementation*.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **OTRIM1[4:0]**: DAC channel1 offset trimming value

27.7.16 DAC mode control register (DAC_MCR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SINFORMAT2	DMADOUBLE2	Res.	Res.	Res.	Res.	Res.	MODE2[2:0]		
						rw	rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HFSEL[2:0]			Res.	Res.	Res.	SINFORMAT1	DMADOUBLE1	Res.	Res.	Res.	Res.	Res.	MODE1[2:0]		
rw	rw	rw				rw	rw						rw	rw	rw

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

Bit 25 **SINFORMAT2**: Enable signed format for DAC channel2

This bit is set and cleared by software.

0: Input data is in unsigned format

1: Input data is in signed format (2's complement). The MSB bit represents the sign.

Note: This bit is available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bit 24 **DMADOUBLE2**: DAC channel2 DMA double data mode

This bit is set and cleared by software.

0: DMA normal mode selected

1: DMA double data mode selected

Note: This bit is available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bits 23:19 Reserved, must be kept at reset value.

Bits 18:16 **MODE2[2:0]:** DAC channel2 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN2 = 0 and bit CEN2 = 0 in the DAC_CR register). If EN2 = 1 or CEN2 = 1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel2 mode:

– DAC channel2 in normal mode

000: DAC channel2 is connected to external pin with Buffer enabled

001: DAC channel2 is connected to external pin and to on chip peripherals with buffer enabled

010: DAC channel2 is connected to external pin with buffer disabled

011: DAC channel2 is connected to on chip peripherals with Buffer disabled

– DAC channel2 in sample and hold mode

100: DAC channel2 is connected to external pin with Buffer enabled

101: DAC channel2 is connected to external pin and to on chip peripherals with Buffer enabled

110: DAC channel2 is connected to external pin and to on chip peripherals with Buffer disabled

111: DAC channel2 is connected to on chip peripherals with Buffer disabled

Note: This register can be modified only when EN2 = 0.

Refer to Section 27.3: DAC implementation for the availability of DAC channel2.

Bits 15:13 **HFSEL[2:0]**: High frequency interface mode selection
000: High frequency interface mode disabled
001: High frequency interface mode enabled for 80 MHz ≤ dac_ker_ck ≤ 160 MHz
010: High frequency interface mode enabled for 160 MHz ≤ dac_ker_ck
Others: Reserved, must not be used.

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 **SINFORMAT1**: Enable signed format for DAC channel1
This bit is set and cleared by software.
0: Input data is in unsigned format
1: Input data is in signed format (2's complement). The MSB bit represents the sign.

Bit 8 **DMADOUBLE1**: DAC channel1 DMA double data mode
This bit is set and cleared by software.
0: DMA normal mode selected
1: DMA double data mode selected

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

Bits 2:0 **MODE1[2:0]:** DAC channel1 mode
These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.
They can be set and cleared by software to select the DAC channel1 mode:
– DAC channel1 in normal mode
000: DAC channel1 is connected to external pin with Buffer enabled
001: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
010: DAC channel1 is connected to external pin with Buffer disabled
011: DAC channel1 is connected to on chip peripherals with Buffer disabled
– DAC channel1 in sample & hold mode
100: DAC channel1 is connected to external pin with Buffer enabled
101: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
110: DAC channel1 is connected to external pin and to on chip peripherals with Buffer disabled
111: DAC channel1 is connected to on chip peripherals with Buffer disabled
Note: This register can be modified only when EN1 = 0.

27.7.17 DAC channel1 sample and hold sample time register (DAC_SHSR1)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 9:0 **TSAMPLE1[9:0]:** DAC channel1 sample time (only valid in sample and hold mode)

These bits can be written when the DAC channel1 is disabled or also during normal operation. in the latter case, the write can be done only when BWST1 of DAC_SR register is low, If BWST1 = 1, the write operation is ignored.

*Note: It represents the number of LSI/LSE clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI/LSE clock period.*

27.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2)

This register is available only on dual-channel DACs. Refer to *Section 27.3: DAC implementation*.

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 9:0 **TSAMPLE2[9:0]:** DAC channel2 sample time (only valid in sample and hold mode)
These bits can be written when the DAC channel2 is disabled or also during normal operation. in the latter case, the write can be done only when BWST2 of DAC_SR register is low, if BWST2 = 1, the write operation is ignored.

*Note: It represents the number of LSI/LSE clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI/LSE clock period.*

27.7.19 DAC sample and hold time register (DAC_SHHR)

Address offset: 0x48

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 25:16 **THOLD2[9:0]:** DAC channel2 hold time (only valid in sample and hold mode).
Hold time = (THOLD[9:0]) x LSI/LSE clock period
These bits are available only on dual-channel DACs. Refer to *Section 27.3: DAC implementation*.
Note: This register can be modified only when EN2 = 0.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **THOLD1[9:0]:** DAC channel1 hold time (only valid in sample and hold mode)
Hold time = (THOLD[9:0]) x LSI/LSE clock period
Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

27.7.20 DAC sample and hold refresh time register (DAC_SHRR)

Address offset: 0x4C

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **TREFRESH2[7:0]:** DAC channel2 refresh time (only valid in sample and hold mode)
Refresh time = (TREFRESH[7:0]) x LSI/LSE clock period
These bits are available only on dual-channel DACs. Refer to *Section 27.3: DAC implementation*.
Note: This register can be modified only when EN2 = 0.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **TREFRESH1[7:0]:** DAC channel1 refresh time (only valid in sample and hold mode)
Refresh time = (TREFRESH[7:0]) x LSI/LSE clock period
Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

27.7.21 DAC channel1 sawtooth register (DAC_STR1)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STINCDATA1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	STEXT1	STDIR1	STRSTDATA1[11:0]											
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **STINCDATA1[15:0]:** DAC channel1 sawtooth increment value (12.4 bit format)

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

Bit 13 **STEXT1**: DAC channel1 sawtooth extended setting
0: Sawtooth reset value defined by STRSTDATA[11:0]
1: Sawtooth reset value defined by STEXTRSTDATA[15:0]

Bit 12 **STDIR1:** DAC channel1 sawtooth direction setting
This bit is written by software to select the direction of sawtooth step direction:
0: Decrement
1: Increment

Bits 11:0 **STRSTDATA1[11:0]:** DAC channel1 sawtooth reset value

27.7.22 DAC channel2 sawtooth register (DAC_STR2)

This register is available only on dual-channel DACs. Refer to *Section 27.3: DAC implementation*.

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STINCDATA2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	STEXT2	STDIR2	STRSTDATA2[11:0]											
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **STINCDATA2[15:0]:** DAC channel2 Sawtooth increment value (12.4 bit format)

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

Bit 13 **STEXT2:** DAC channel2 sawtooth extended setting
0: Sawtooth reset value defined by STRSTDATA[11:0]
1: Sawtooth reset value defined by STEXTRSTDATA[15:0]

Bit 12 **STDIR2:** DAC channel2 sawtooth direction setting
This bit is written by software to select the direction of sawtooth step direction
0: Decrement
1: Increment

Bits 11:0 **STRSTDATA2[11:0]:** DAC channel2 sawtooth reset value

27.7.23 DAC sawtooth mode register (DAC_STMODR)

Address offset: 0x60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	STINCTRIGSEL2[4:0]					Res.	Res.	Res.	STRSTTRIGSEL2[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	STINCTRIGSEL1[4:0]					Res.	Res.	Res.	STRSTTRIGSEL1[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **STINCTRIGSEL2[4:0]:** DAC channel2 sawtooth increment trigger (STINCTRIG) selection
For details on trigger configuration and mapping, refer to *Section 27.4.2: DAC pins and internal signals*.
00000: SWTRIGB2
00001: dac_inc_ch2_trg1
....
11111: dac_inc_ch2_trg31

Note: These bits are available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **STRSTTRIGSEL2[4:0]:** DAC channel2 sawtooth reset trigger (STRSTTRIG) selection
For details on trigger configuration and mapping, refer to *Section 27.4.2: DAC pins and internal signals*.
00000: SWTRIGB2
00001: dac_ch2_trg1
....
11111: dac_ch2_trg31
The mapping is the same as for TSEL2[4:0].

Note: These bits are available only on dual-channel DACs. Refer to Section 27.3: DAC implementation.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **STINCTRIGSEL1[4:0]:** DAC channel1 sawtooth increment trigger (STINCTRIG) selection
For details on trigger configuration and mapping, refer to *Section 27.4.2: DAC pins and internal signals*.
00000: SWTRIGB1
00001: dac_inc_ch1_trg1
....
11111: dac_inc_ch1_trg31

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **STRSTTRIGSEL1[4:0]:** DAC channel1 sawtooth reset trigger (STRSTTRIG) selection
For details on trigger configuration and mapping, refer to *Section 27.4.2: DAC pins and internal signals*.
00000: SWTRIG1
00001: dac_ch1_trg1
....
11111: dac_ch1_trg31
The mapping is the same as for TSEL1[4:0].

27.7.24 DAC extended sawtooth reset control register (DAC_STEXTRST)

Address offset: 0x64

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STEXTRSTDATA2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STEXTRSTDATA1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **STEXTRSTDATA2[15:0]:** DAC channel2 extended sawtooth reset value
This bitfield contains the extended sawtooth value programmed at reset, in 2's complement signed format.

Bits 15:0 **STEXTRSTDATA1[15:0]:** DAC channel1 extended sawtooth reset value
This bitfield contains the extended sawtooth value programmed at reset, in 2's complement signed format.

27.7.25 DAC extended sawtooth minimum and maximum value register (DAC_STEXTMM1)

Address offset: 0x68

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	STEXTMAX1[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	STEXTMIN1[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **STEXTMAX1[11:0]:** DAC channel1 extended sawtooth maximum value
This bitfield contains the extended sawtooth maximum out value.

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

Bits 11:0 **STEXTMIN1[11:0]:** DAC channel1 extended sawtooth minimum value
This bitfield contains the extended sawtooth minimum out value.

27.7.26 DAC extended sawtooth minimum and maximum value register (DAC_STEXTMM2)

Address offset: 0x6C

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	STEXTMAX2[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	STEXTMIN2[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **STEXTMAX2[11:0]:** DAC channel2 extended sawtooth maximum value
This bitfield contains the extended sawtooth maximum out value.

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

Bits 11:0 **STEXTMIN2[11:0]:** DAC channel2 extended sawtooth minimum value
This bitfield contains the extended sawtooth minimum out value.

27.7.27 DAC option register (DAC_OR)

Address offset: 0x3EC

Reset value: 0x0000 0000

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

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

Bits 3:2 **OP[3:2]:** Option bits
For DAC2, 3, 4
00: Output glitch control disabled for dac_out2 channel
01: Output glitch control forced to the reverse direction level for dac_out2 channel
10: Output glitch control forced to the forward direction for dac_out2 channel
11: Reserved, must not be used.

Bits 1:0 **OP[1:0]:** Option bits
For DAC2, 3, 4
00: Output glitch control disabled for dac_out1 channel
01: Output glitch control forced to the reverse direction for dac_out1 channel
10: Output glich control forced to the forward direction for dac_out1 channel
11: Reserved, must not be used.

27.7.28 DAC register map

Table 272 summarizes the DAC registers.

Table 272. DAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	TSEL2 [4]	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[2:0]		TSEL2[3:0]				TEN2	EN2	TSEL1[4]	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3:0]				TEN1	EN1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	DAC_ SWTRGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIGB2	SWTRIGB1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
	Reset value															0	0															0	0
0x08	DAC_ DHR12R1	Res.	Res.	Res.	Res.	DACC1DHRB[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x0C	DAC_ DHR12L1	DACC1DHRB[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x10	DAC_ DHR8R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHRB[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	DAC_ DHR12R2	Res.	Res.	Res.	Res.	DACC2DHRB[11:0]												Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x18	DAC_ DHR12L2	DACC2DHRB[11:0]												Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x1C	DAC_ DHR8R2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHRB[7:0]								DACC2DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	DAC_ DHR12RD	Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x24	DAC_ DHR12LD	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				

Table 272. DAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	DAC_DHR8RD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	DAC_DOR1	Res.	Res.	Res.	Res.	DACC1DORB[11:0]												Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x30	DAC_DOR2	Res.	Res.	Res.	Res.	DACC2DORB[11:0]												Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x34	DAC_SR	BWST2	CAL_FLAG2	DMAUDR2	DORSTAT2	DAC2RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BWST1	CAL_FLAG1	DMAUDR1	DORSTAT1	DAC1RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0												0	0	0	0	0											
0x38	DAC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4]	OTRIM2[3]	OTRIM2[2]	OTRIM2[1]	OTRIM2[0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4]	OTRIM1[3]	OTRIM1[2]	OTRIM1[1]	OTRIM1[0]
	Reset value												X	X	X	X	X												X	X	X	X	X
0x3C	DAC_MCR	Res.	Res.	Res.	Res.	Res.	Res.	SINFORMAT2	DMADOUBLE2	Res.	Res.	Res.	Res.	Res.	MODE2[2:0]			HFSEL[2]	HFSEL[1]	HFSEL[0]	Res.	Res.	Res.	SINFORMAT1	DMADOUBLE1	Res.	Res.	Res.	Res.	Res.	MODE1[2:0]		
	Reset value							0	0						0	0	0	0	0	0				0	0						0	0	0
0x40	DAC_SHSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x44	DAC_SHSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x48	DAC_SHHR	Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	1							0	0	0	0	0	0	0	0	0	1
0x4C	DAC_SHRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
	Reset value									0	0	0	0	0	0	0	1									0	0	0	0	0	0	0	1
0x50-0x54	Reserved																																
0x58	DAC_STR1	STINCDATA1[15:0]																Res.	Res.	STEXT1	STDIR1	STRSTDATA1[11:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	DAC_STR2	STINCDATA2[15:0]																Res.	Res.	STEXT2	STDIR2	STRSTDATA2[11:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 272. DAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x60	DAC_ STMODR	Res.	Res.	Res.	STINCTRIGSEL2[4]	STINCTRIGSEL2[3]	STINCTRIGSEL2[2]	STINCTRIGSEL2[1]	STINCTRIGSEL2[0]	Res.	Res.	Res.	STRSTTRIGSEL2[4]	STRSTTRIGSEL2[3]	STRSTTRIGSEL2[2]	STRSTTRIGSEL2[1]	STRSTTRIGSEL2[0]	Res.	Res.	Res.	STINCTRIGSEL1[4]	STINCTRIGSEL1[3]	STINCTRIGSEL1[2]	STINCTRIGSEL1[1]	STINCTRIGSEL1[0]	Res.	Res.	Res.	STRSTTRIGSEL1[4]	STRSTTRIGSEL1[3]	STRSTTRIGSEL1[2]	STRSTTRIGSEL1[1]	STRSTTRIGSEL1[0]
	Reset value				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x64	DAC_ST EXTRST	STEXTRSTDATA2[15:0]																STEXTRSTDATA1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68	DAC_ST EXTMM1	Res.	Res.	Res.	Res.	STEXTMAX1[11:0]												Res.	Res.	Res.	Res.	STEXTMIN1[11:0]											
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x6C	DAC_ST EXTMM2	Res.	Res.	Res.	Res.	STEXTMAX2[11:0]												Res.	Res.	Res.	Res.	STEXTMIN2[11:0]											
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x3EC	DAC_OR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OP3	OP2	OP1	OP0
	Reset value																													0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

28 Voltage reference buffer (VREFBUF)

28.1 VREFBUF introduction

The devices embed a voltage reference buffer which can be used as voltage reference for the on-chip ADCs, and also as voltage reference for external components through the VREF+ pin.

28.2 VREFBUF implementation

The table below describes the VREFBUF voltage typical values:

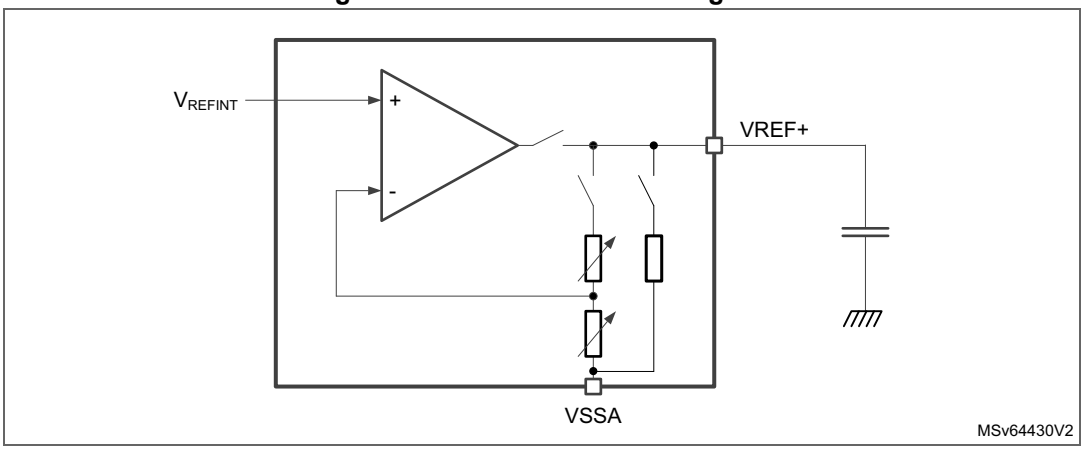
Table 273. VREFBUF typical values

Symbol	Value
$V_{REFBUF0}$	2.9 V
$V_{REFBUF1}$	2.5 V
$V_{REFBUF2}$	2.048 V
$V_{REFBUF3}$	1.8 V

Note: Refer to the product datasheet for more details.

28.3 VREFBUF functional description

Figure 376. VREFBUF block diagram



The internal voltage reference buffer is an operational amplifier, with programmable gain. The amplifier input is connected to the internal voltage reference V_{REFINT}. The VREFBUF supports four voltages[a], which are configured with VRS bits in the VREFBUF_CSR register:

- VRS = 000: $V_{REFBUF0}$ voltage selected
- VRS = 001: $V_{REFBUF1}$ voltage selected
- VRS = 010: $V_{REFBUF2}$ voltage selected
- VRS = 011: $V_{REFBUF3}$ voltage selected

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:

Table 274. VREF buffer modes

ENVR	HIZ	VREF buffer configuration
0	0	VREFBUF buffer off mode: – VREF+ pin pulled-down to V_{SSA}
0	1	External voltage reference mode (default value): – VREFBUF buffer off – VREF+ pin input mode
1	0	Internal voltage reference mode: – VREFBUF buffer on – VREF+ pin connected to VREFBUF buffer output
1	1	Hold mode: – VREF is enable without output buffer, VREF+ pin voltage is hold with the external capacitor – VRR detection disabled and VRR bit keeps last state

After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

a. The minimum V_{DDA} voltage depends on VRS setting, refer to the product datasheet.

28.4 VREFBUF trimming

The VREFBUF output voltage is factory-calibrated by ST. At reset, the calibration data for the VRS=0 setting is automatically loaded to the TRIM register. If the VRS setting is changed, the software must copy the trimming data from the read-only system memory area (flash memory) to the TRIM[5:0] field of the VREFBUF_CCR register.

The system memory address of the trimming data for each VRS voltage setting is given in *Table 275*. Software must copy the six LSBs of data from the corresponding address to the TRIM[5:0] field when the VRS setting is changed.

Optionally, the user can trim the output voltage by changing the TRIM[5:0] register bits directly.

Table 275. VREFBUF trimming data

Calibration value name	Description	Memory address
V_{REF_VS0}	VREFBUF trimming value for VRS=0	0x5200 2435
V_{REF_VS1}	VREFBUF trimming value for VRS=1	0x08FF F81C
V_{REF_VS2}	VREFBUF trimming value for VRS=2	0x08FF F820
V_{REF_VS3}	VREFBUF trimming value for VRS=3	0x08FF F824

28.5 VREFBUF registers

28.5.1 VREFBUF control and status register (VREFBUF_CSR)

Address offset: 0x00

Reset value: 0x0000 0002

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

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **VRS[2:0]**: Voltage reference scale

These bits select the value generated by the voltage reference buffer.

VRS = 000: $V_{REFBUF0}$ voltage selected.
VRS = 001: $V_{REFBUF1}$ voltage selected.
VRS = 010: $V_{REFBUF2}$ voltage selected.
VRS = 011: $V_{REFBUF3}$ voltage selected.
Others: Reserved

Note: Refer to the product datasheet for each $V_{REFBUFx}$ voltage setting value.
The software can program this bitfield only when the VREFBUF is disabled (ENVR=0).

Bit 3 **VRR:** Voltage reference buffer ready

0: the voltage reference buffer output is not ready.
1: the voltage reference buffer output reached the requested level.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HIZ:** High impedance mode

This bit controls the analog switch to connect or not the VREF+ pin.
0: VREF+ pin is internally connected to the voltage reference buffer output.
1: VREF+ pin is high impedance.
Refer to *Table 274: VREF buffer modes* for the mode descriptions depending on ENVR bit configuration.

Bit 0 **ENVR:** Voltage reference buffer mode enable

This bit is used to enable the voltage reference buffer mode.
0: Internal voltage reference mode disable (external voltage reference mode).
1: Internal voltage reference mode (reference buffer enable or hold mode) enable.

28.5.2 VREFBUF calibration control register (VREFBUF_CCR)

Address offset: 0x04

Reset value: 0x0000 00XX

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

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

Bits 5:0 **TRIM[5:0]**: Trimming code

These bits are automatically initialized after reset with the trimming value for VRS=0 stored in the flash memory during the production test.

When the VRS value is changed, the software must copy the trimming data from the read-only system memory area (flash memory) to the TRIM[5:0] bitfield. Writing into these bits allows the tuning of the internal reference buffer voltage. If VRS is reset to 0 from another value, the trimming data for VRS=0 must be copied again.

Note: If the user application performs the trimming, the trimming code must start from 000000 to 111111 in ascending order.

28.5.3 VREFBUF register map

The following table gives the VREFBUF register map and the reset values.

Table 276. VREFBUF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	VREFBUF_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRS[2:0]			VRR	Res.	HIZ	ENVR
	Reset value																										0	0	0	0		1	0
0x04	VREFBUF_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
	Reset value																											X	X	X	X	X	X

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

29 Comparator (COMP)

29.1 COMP introduction

The comparator can be used for a variety of functions including:

- Wake up from a low-power mode triggered by an analog signal
- Analog signal conditioning
- Cycle-by-cycle current control loop when combined with a PWM output from a timer

29.2 COMP main features

- Selectable inverting analog inputs:
  - Multiplexed I/O pins
  - DAC channels
  - Internal reference voltage and three submultiple values (1/4, 1/2, 3/4) provided by the scaler (buffered voltage divider)
- Mapping of outputs to I/Os
- Interrupt generation capability with wake up from Sleep and Stop modes (through the EXTI controller)
- Direct interrupt output to the CPU
- Programmable speed/power mode
- Programmable hysteresis
- Programmable digital filtering
- Redirection of outputs to timer inputs for triggering:
  - Capture events
  - Cycle-by-cycle current control
  - Break events for fast PWM shutdowns
- Output blanking for immunity to switching noise
- Window paired comparator
- Deglitcher

29.3 Implementation

Table 277. COMP implementation(1)

COMP features	COMP1	COMP2	COMP3	COMP4	COMP5	COMP6	COMP7	COMP8
Window mode	X	X	X	X	X	X	X	X
Hysteresis	X	X	X	X	X	X	X	X
Power/speed	-	-	-	-	-	-	-	-
Deglitcher	X	X	X	X	X	X	X	X
Digital filtering	X	X	X	X	X	X	X	X
Interrupt through NVIC	X	X	X	X	X	X	X	X
Option register	-	-	-	-	-	-	-	-

1. 'X' = supported, '-' = not supported.

The following table lists the paired comparators required for the window mode.

Table 278. Window paired comparators

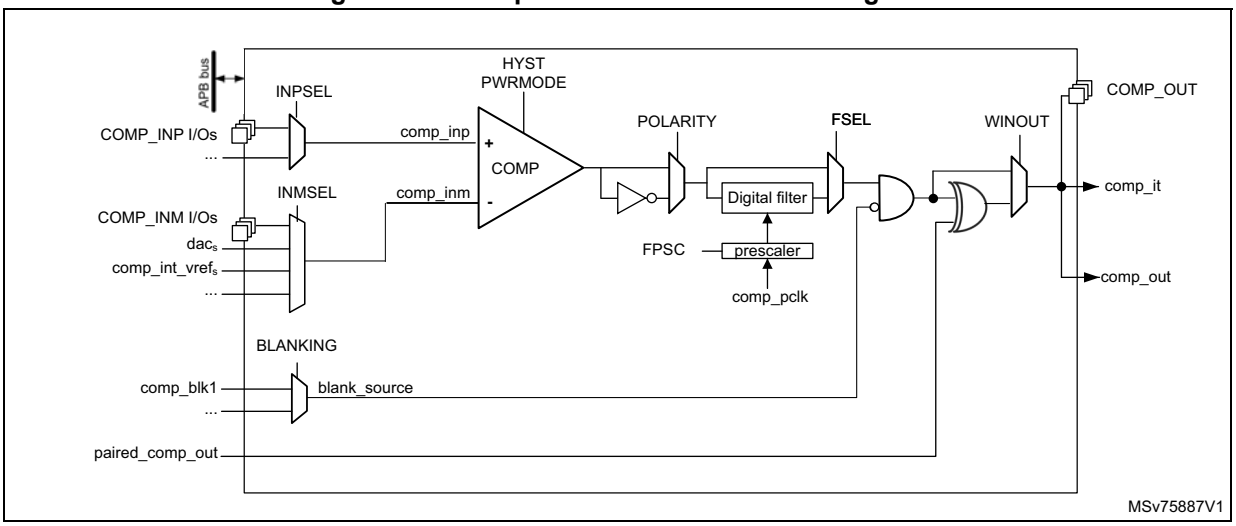
Window comparator	#1	#2	#3	#4	#5	#6	#7	#8
Upper threshold comparator	COMP1	COMP2	COMP3	COMP4	COMP5	COMP6	COMP7	COMP8
Lower threshold comparator	COMP2	COMP1	COMP4	COMP3	COMP6	COMP5	COMP8	COMP7

29.4 COMP functional description

29.4.1 COMP block diagram

The block diagram of the comparators is shown in *Figure 377*.

Figure 377. Comparator functional block diagram



29.4.2 COMP pins and internal signals

The I/Os used as comparator inputs must be configured in analog mode in the GPIO registers.

The comparator outputs can be connected to the I/Os through their alternate functions (refer to the product datasheet).

The outputs can also be internally redirected to a variety of timer inputs for the following purposes:

- Emergency shut-down of PWM signals, using BKIN and BKIN2 inputs
- Cycle-by-cycle current control, using ETR inputs of timers
- Input capture for timing measurements

The comparator output can be routed simultaneously internally and to the I/O pins.

Table 279. COMP input/output pins

Pin name	Signal type	Description
VDDA	Input, analog supply	Analog power supply and positive reference voltage for the comparator
VSSA	Input, analog supply ground	Ground for analog power supply, equal to VSS.
COMP_INMy/INPy	external analog input signal	Comparator Inverting and Non-inverting analog input channels
COMP_OUT	external digital output signal	Comparator digital output

Table 280. COMP input/output internal signals

Signal name	Signal type	Description
comp_inm$_y$	Analog inputs	Inverting COMP input sources
comp_inp$_y$	Analog inputs	Non-inverting COMP input sources
comp_blk$_x$	Digital inputs	COMP blanking sources
comp_pclk	Digital input	COMP peripheral clock
comp_out	Digital output	COMP output
comp_it	Digital output	COMP interrupt
dac	Analog inputs	internal dac signals
comp_int_vref	Analog inputs	comparator internal reference signals from scaler
paired_comp_out	Digital input	window paired comparator signal

Table 281. COMP_INP non-inverting input signals (COMP1, COMP2, COMP3, COMP4)

INPSEL [1:0]	COMP1	COMP2	COMP3	COMP4
00	COMP1_INP1	COMP2_INP1	COMP3_INP1	COMP4_INP1
01	COMP1_INP2	COMP2_INP2	COMP3_INP2	COMP4_INP2
10	COMP1_INP3	COMP2_INP3	COMP3_INP3	COMP4_INP3
11	comp2_inp(1)	comp1_inp(1)	comp4_inp(1)	comp3_inp(1)

1. Window paired comparator signal.

Table 282. COMP_INP non-inverting input signals (COMP5, COMP6, COMP7, COMP8)

INPSEL [1:0]	COMP5	COMP6	COMP7	COMP8
00	COMP5_INP1	COMP6_INP1	COMP7_INP1	COMP8_INP1
01	COMP5_INP2	COMP6_INP2	COMP7_INP2	COMP8_INP2
10	COMP5_INP3	COMP6_INP3	COMP7_INP3	COMP8_INP3
11	comp6_inp(1)	comp5_inp(1)	comp8_inp(1)	comp7_inp(1)

1. Window paired comparator input.

Table 283. COMP_INM inverting input signals (COMP1, COMP2, COMP3, COMP4)

INMSEL [3:0]	COMP1	COMP2	COMP3	COMP4
0000	1/4 comp_int_vref	1/4 comp_int_vref	1/4 comp_int_vref	1/4 comp_int_vref
0001	1/2 comp_int_vref	1/2 comp_int_vref	1/2 comp_int_vref	1/2 comp_int_vref
0010	3/4 comp_int_vref	3/4 comp_int_vref	3/4 comp_int_vref	3/4 comp_int_vref
0011	comp_int_vref	comp_int_vref	comp_int_vref	comp_int_vref
0100	dac1_out1	dac1_out2	-	-

Table 283. COMP_INM inverting input signals (COMP1, COMP2, COMP3, COMP4)

INMSEL [3:0]	COMP1	COMP2	COMP3	COMP4
0101	dac3_out2	dac4_out1	dac4_out1	dac4_out2
0110	-	-	dac1_out1	dac1_out1
0111	COMP1_INM1	COMP2_INM1	COMP3_INM1	COMP4_INM1
1000	COMP1_INM2	COMP2_INM2	COMP3_INM2	COMP4_INM2
1001	comp2_inm(1)	comp1_inm(1)	comp4_inm(1)	comp3_inm(1)

1. Window paired comparator signal.

Table 284. COMP_INM inverting input signals (COMP5, COMP6, COMP7, COMP8)

INMSEL [3:0]	COMP5	COMP6	COMP7	COMP8
0000	1/4 comp_int_vref	1/4 comp_int_vref	1/4 comp_int_vref	1/4 comp_int_vref
0001	1/2 comp_int_vref	1/2 comp_int_vref	1/2 comp_int_vref	1/2 comp_int_vref
0010	3/4 comp_int_vref	3/4 comp_int_vref	3/4 comp_int_vref	3/4 comp_int_vref
0011	comp_int_vref	comp_int_vref	comp_int_vref	comp_int_vref
0100	dac2_out1	dac2_out2	dac3_out1	dac3_out2
0101	dac1_out2	-	-	-
0110	dac4_out2	dac3_out2	dac2_out2	dac2_out1
0111	COMP5_INM1	COMP6_INM1	COMP7_INM1	COMP8_INM1
1000	COMP5_INM2	COMP6_INM2	COMP7_INM2	COMP8_INM2
1001	comp6_inm(1)	comp5_inm(1)	comp8_inm(1)	comp7_inm(1)

1. Window paired comparator signal.

Table 285. COMP blanking internal input signals (COMP1, COMP2, COMP3, COMP4)

BLANKING[3:0]	COMP1	COMP2	COMP3	COMP4
0000	No blanking	No blanking	No blanking	No blanking
0001	tim1_oc5	tim1_oc5	tim1_oc5	tim1_oc5
0010	tim1_oc7	tim1_oc7	tim1_oc6	tim1_oc6
0011	tim2_oc3	tim2_oc3	tim2_oc3	tim2_oc3
0100	tim3_oc3	tim3_oc3	tim3_oc3	tim3_oc3
0101	tim4_oc3	tim4_oc3	tim4_oc3	tim4_oc3
0110	tim8_oc5	tim8_oc5	tim8_oc5	tim8_oc5
0111	tim8_oc7	tim8_oc7	tim8_oc6	tim8_oc6
1000	tim12_oc1	tim12_oc1	tim12_oc1	tim12_oc1
1001	tim20_oc5	tim20_oc5	tim20_oc5	tim20_oc5
1010	tim20_oc7	tim20_oc7	tim20_oc6	tim20_oc6

Table 285. COMP blanking internal input signals (COMP1, COMP2, COMP3, COMP4)

BLANKING[3:0]	COMP1	COMP2	COMP3	COMP4
1011	tim15_oc1	tim15_oc1	tim15_oc1	tim15_oc1
1100	tim15_oc2	tim15_oc2	tim15_oc2	tim15_oc2
1101	-	-	-	-
1110	tim9_oc1	tim9_oc1	-	-
1111	-	-	-	-

Table 286. COMP blanking internal input signals (COMP5, COMP6, COMP7, COMP8)

BLANKING [3:0]	COMP5	COMP6	COMP7	COMP8
0000	No blanking	No blanking	No blanking	No blanking
0001	tim1_oc5	tim1_oc5	tim1_oc5	tim1_oc5
0010	tim1_oc7	tim1_oc7	tim1_oc6	tim1_oc6
0011	tim2_oc4	tim2_oc4	tim9_oc1	tim9_oc1
0100	tim3_oc3	tim3_oc3	tim3_oc4	tim3_oc4
0101	tim4_oc3	tim4_oc3	tim4_oc3	tim4_oc3
0110	tim8_oc5	tim8_oc5	tim8_oc5	tim8_oc5
0111	tim8_oc7	tim8_oc7	tim8_oc6	tim8_oc6
1000	tim12_oc1	tim12_oc1	tim12_oc1	tim12_oc1
1001	tim20_oc5	tim20_oc5	tim20_oc5	tim20_oc5
1010	tim20_oc7	tim20_oc7	tim20_oc6	tim20_oc6
1011	tim15_oc1	tim15_oc1	tim15_oc1	tim15_oc1
1100	tim15_oc2	tim15_oc2	tim15_oc2	tim15_oc2
1101	-	-	-	-
1110	-	-	-	-
1111	-	-	-	-

29.4.3 COMP startup stabilization time

The comparator parameters and its output are undefined after the COMP enable (power-on) for a period defined as startup stabilization time, see the product datasheet for more details.

If the comparator output is driving any critical stage of the design, it is recommended to redirect the comparator output to the device I/O through the GPIO alternate function or through IP configuration (such as timers) after the stabilization time elapses.

29.4.4 COMP power and speed modes

When the comparator has the power/speed feature, PWRMODE[1:0] is used to adjust power consumption versus propagation delay (refer to product datasheet) to have the optimum trade-off for a given application.

29.4.5 COMP reset and clocks

The clock comp_pclk provided by the clock controller is synchronous with the APB clock.

Note: ***Important:*** *The polarity selection logic and the output redirection to the port work independently from the APB clock. This allows the comparator to work even in Stop mode. The interrupt line, connected to the NVIC of the CPU, requires the APB clock (comp_pclk) to work. In the absence of the APB clock, the interrupt signal comp_it cannot be generated.*

29.4.6 COMP LOCK mechanism

The comparators can be used for safety purposes, such as overcurrent or thermal protection. For applications with specific functional safety requirements, the comparator configuration can be protected against undesired alteration that could happen, for example, at program counter corruption.

For this purpose, the comparator configuration registers can be write-protected (read-only).

Upon configuring a comparator, its LOCK bit is set to 1. This causes the whole register set of the comparator to become read only, including the LOCK bit itself.

The write protection can only be removed through the MCU reset.

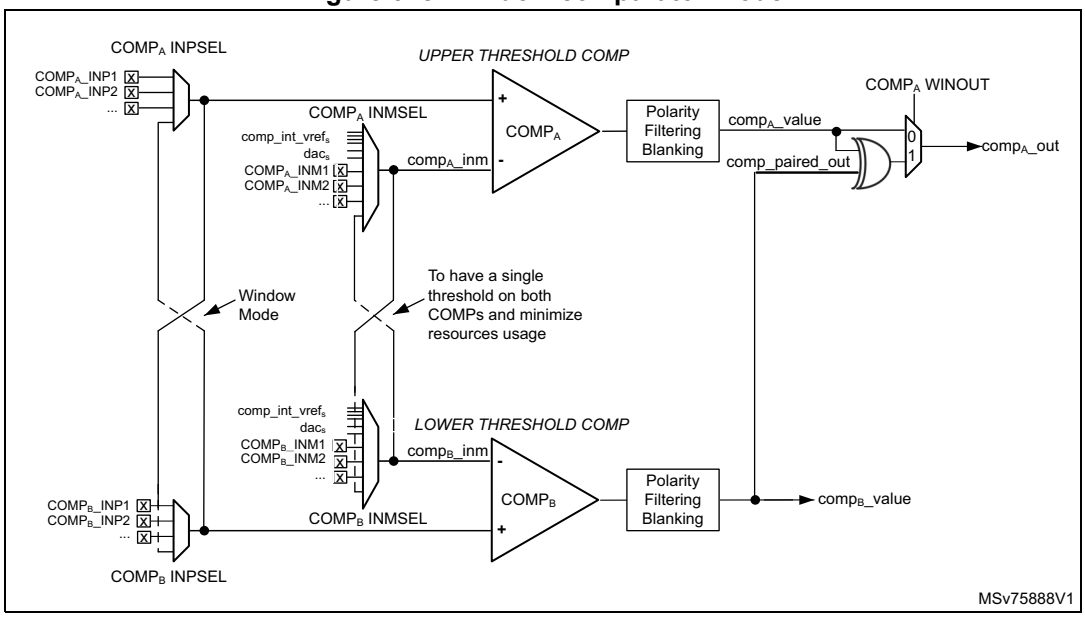
29.4.7 COMP window mode

When the feature is available, the window mode uses two paired comparators ($COMP_A$ and paired $COMP_B$) to create a window comparator (refer to *Table 278: Window paired comparators*). It can be used to monitor if the analog voltage is within the voltage range defined by the lower and upper threshold,

The monitored analog voltage is connected to the noninverting inputs of the paired comparators. The upper and lower threshold voltages are connected to the inverting inputs of the comparators to define the window voltage.

The output of the window comparator is selected with the control bit WINOUT.

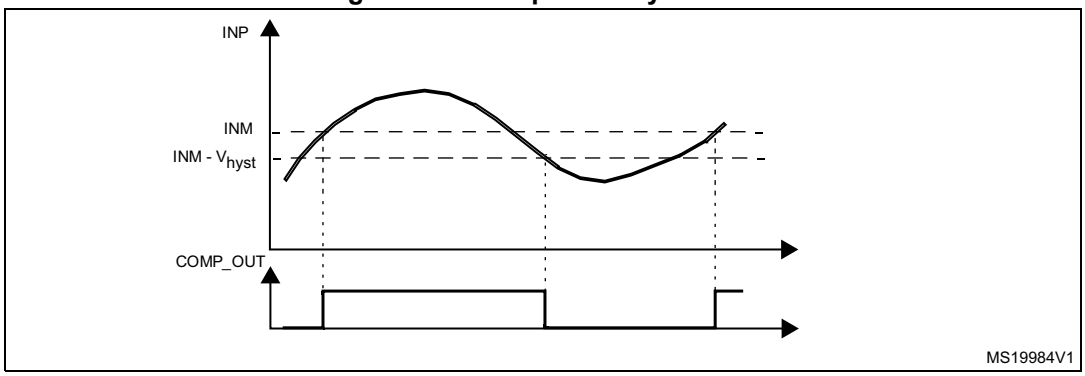
Figure 378. Window comparator mode



29.4.8 COMP hysteresis

When the hysteresis feature is available, the HYST[2:0] bitfield can be programmed to select the hysteresis voltage and avoid spurious output transitions in case of noisy signals. The hysteresis can be disabled if it is not needed.

Figure 379. Comparator hysteresis



29.4.9 COMP deglitcher

When the deglitcher feature is available, it complements the programmable hysteresis: the hysteresis filters a voltage amplitude, the deglitcher filters high frequency noise on the comparator input signal. The embedded deglitcher analog feature is enabled with the DGLT control bit. Use of the deglitcher increases the comparator propagation delay by a few nanoseconds (refer to product datasheet).

High-speed comparators are highly sensitive to noise. As a result, the comparator output can oscillate at low input signal slopes, even with hysteresis enabled. The deglitcher is an analog function that removes these oscillations from the comparator output. The advantage is that the solution is purely analog: no need of extra digital circuitry and high-speed clock, no extra latency due to digital filtering, negligible power consumption. It is recommended to use the deglitcher coupled with hysteresis.

29.4.10 COMP digital signal filter

When the digital signal filter feature is available, it complements the programmable hysteresis and the deglitcher. The comparator low frequency noise on input signal can be filtered with the digital signal filter. Based on an internal counter clocked by filter_clk (divided from comp_pclk): the output of the comparator is filtered based on the assumption that the logic level is stable for a programmable number of consecutive periods. If the input value changes before the counter has reached the value N, the counter is reset and the transition is discarded (considered as a spurious event).

If the counter reaches N, the transition is considered as valid and transmitted as a correct external event. Consequently, the digital filter adds a latency to the external events being filtered, depending on the sampling clock and on the filter length (number of valid samples expected). In the FLTR register, FSEL[3:0] register allows the selection of various filter_clk divider ratios and N value while FPSC[1:0] register controls the prescaler ratio from comp_pclk to filter_clk.

Figure 380. Comparator output digital filter (N=4)

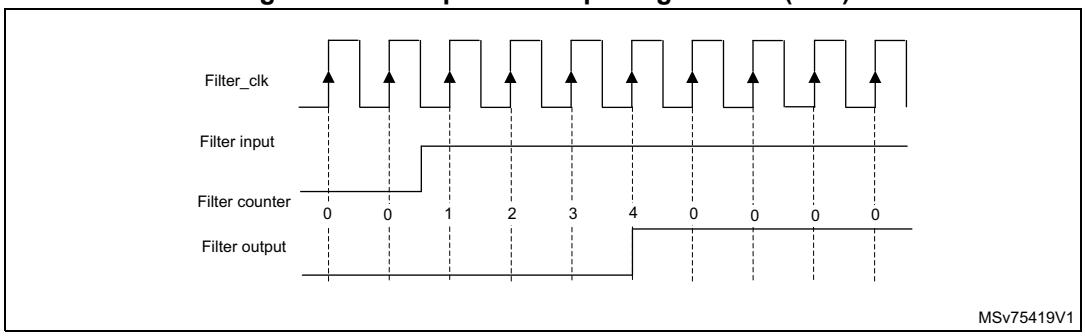
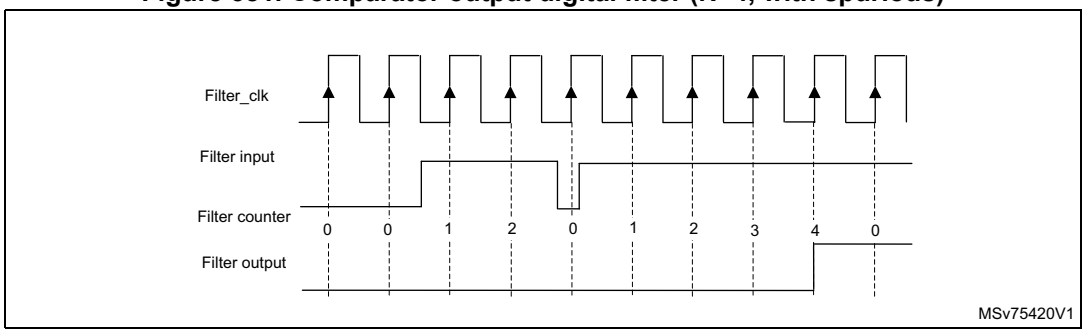


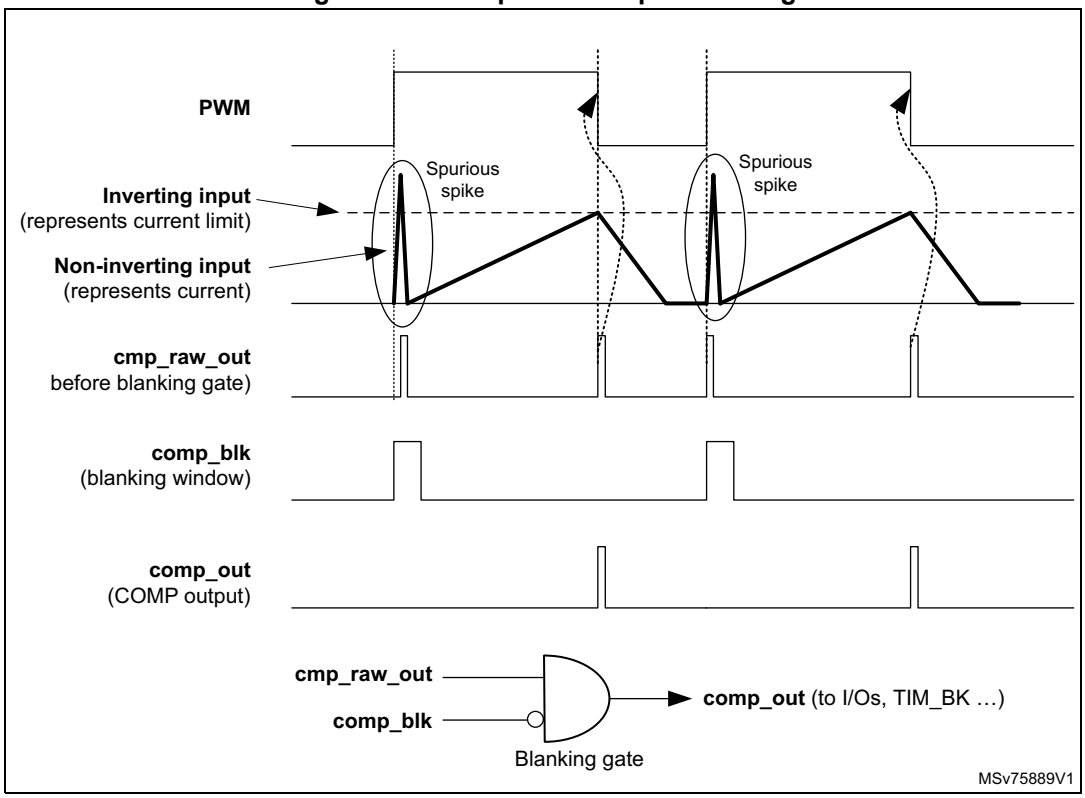
Figure 381. Comparator output digital filter (N=4, with spurious)



29.4.11 COMP output blanking function

The purpose of the blanking function is to prevent the current regulation from being triggered upon short current spikes at the beginning of the PWM period (typically the recovery current in power switches antiparallel diodes). It uses a blanking window defined with a timer output compare signal. Refer to the register description for selectable blanking signals. The blanking signal gates the internal comparator output to clean the comp_out of spurious pulses due to current spikes, as depicted in *Figure 382*.

Figure 382. Comparator output blanking



29.4.12 COMP option register

When the option register feature is available, it can be used for various controls like comparator output mapping or interconnect with other peripherals such as touch sensing. Refer to the register description for detailed information.

29.5 COMP low-power modes

Table 287. Comparator behavior in the low-power modes

Mode	Description
Sleep	No effect on the comparators. Comparator interrupts cause the device to exit the Sleep mode(1).
Stop	No effect on the comparators. Comparator interrupts cause the device to exit the Stop mode(1).
Standby	The COMP registers are powered down and must be reinitialized after exiting.

1. The comparators cannot be used to exit the device from Sleep or Stop mode when the internal reference voltage is switched off.

29.6 COMP interrupts

There are two ways to use the comparator as an interrupt source.

The comparator outputs are internally connected to the extended interrupt and event controller. Each comparator has its own EXTI line and can generate either interrupts or events to make the device exit low-power modes.

When the comparator has the interrupt through NVIC feature, it also provides an interrupt line to the NVIC of the CPU. This functionality is used when the CPU is active to handle low latency interrupt. It requires the APB clock running.

29.6.1 Interrupt through the EXTI block

Refer to *Section 15: Nested vectored interrupt controller (NVIC)* and *Section 16: Extended interrupt and event controller (EXTI)* for more details.

Sequence to enable the COMP interrupt through EXTI block:

1. Enable the COMPx and wait for stabilization.
2. Configure the EXTI line, receiving the compx_wkup signal, in interrupt mode, select the rising, falling, or either-edge sensitivity, and enable the EXTI line.
3. Configure and enable the NVIC IRQ channel mapped to the corresponding EXTI lines.

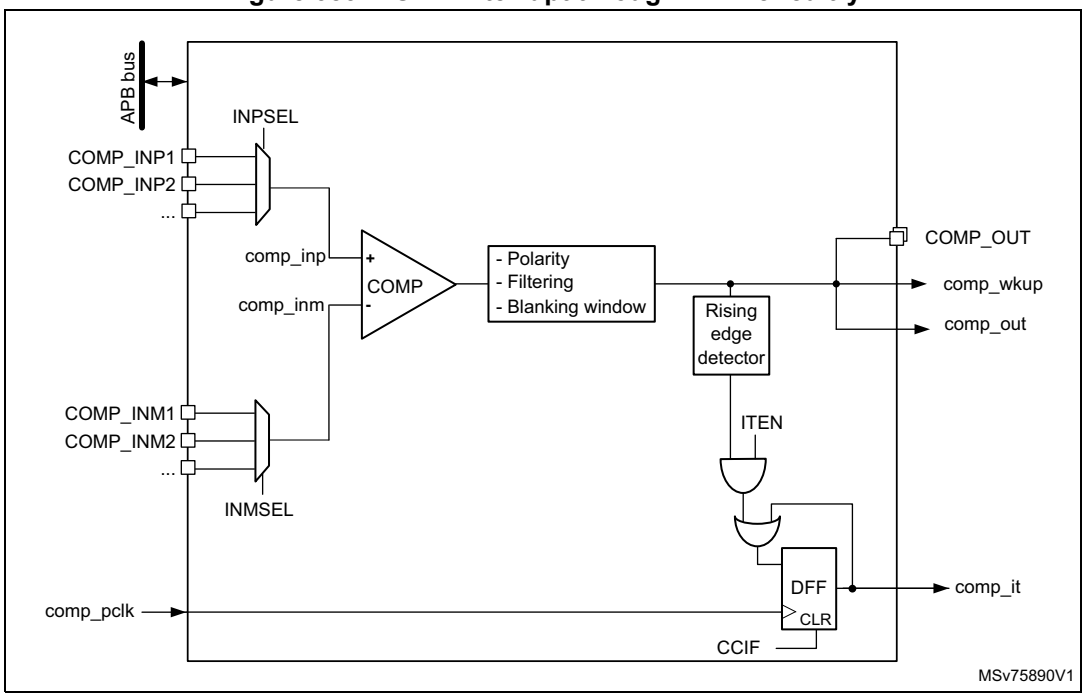
Table 288. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Exit from Sleep mode	Exit from Stop modes	Exit from Standby mode
comp_wkup	through EXTI	through EXTI	Yes	Yes	N/A

29.6.2 Direct interrupt

Refer to *Section 15: Nested vectored interrupt controller (NVIC)* for more details.

Figure 383. COMP interrupt through NVIC circuitry



Sequence to enable the COMP direct interrupt to NVIC of the CPU:

1. Enable the COMP and wait for stabilization.
2. Configure and enable the NVIC IRQ channel mapped to the comp_it line.
3. Configure and enable the ITEN in COMP_CFGRx.
4. The comp_it signal is set when the rising edge is detected from the comparator output.

Table 289. Interrupt control bits

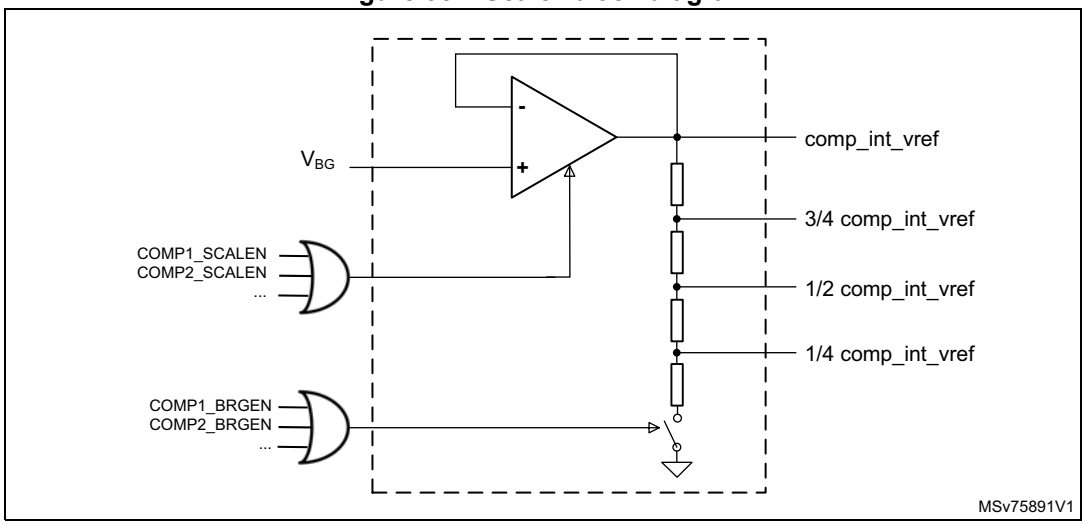
Interrupt event	Interrupt flag	Enable control bit	Interrupt clear bit	Exit from Sleep mode	Exit from Stop modes
comp_it	CIF in COMP_SR	ITEN in COMP_CFGR	CCIF in COMP_IFCR	Yes (With APB clock)	no

Note: It is mandatory to enable the APB clock to use this interrupt. If the clock is not enabled, an interrupt is not generated.

29.7 SCALER function

The scaler block is available to provide the different voltage reference levels to the comparator input. It is based on an amplifier driving a resistor bridge. The amplifier input V_{BG} is connected to an internal voltage reference (refer to product datasheet).

Figure 384. Scaler block diagram



1. The scaler and amplifier is common for all comparators and their enable control bits are logical OR with all comparators.

The amplifier and the resistor bridge can be enabled separately. The amplifier is enabled by the SCALEN bit of the COMP_CFGR register. The resistor bridge is enabled by the BRGEN bit of the COMP_CFGR register.

If SCALEN and BRGEN are set, the four scaler outputs provide the following levels:

- comp_int_vref
- 3/4 comp_int_vref
- 1/2 comp_int_vref
- 1/4 comp_int_vref

If SCALEN is set and BRGEN is reset, all four scaler outputs provide the same level comp_int_vref.

When the resistor divided voltage is not used, the resistor bridge can be disconnected in order to reduce the consumption. When it is disconnected, the four scaler outputs provide the same level comp_int_vref.

29.8 COMP registers

29.8.1 Comparator status register (COMP_SR)

The COMP_SR is the comparator status register.

Address offset: 0x00

Reset value: 0x0000 0000

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

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **CIF**: COMP interrupt flag

This bit is set by hardware when the COMP output is set.
This bit is cleared by software writing 1 to the CCIF bit in the COMP_IFCR register.

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **CVAL:** COMP output status bit

This bit is read-only. It reflects the current COMP output taking into account POLARITY and BLANKING bits effect.

29.8.2 Comparator interrupt clear flag register (COMP_ICFR)

The COMP_ICFR is the comparator interrupt clear flag register.

Address offset: 0x04

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCIF
															rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **CCIF**: Clear COMP interrupt flag

Writing 1 clears the CIF flag in the COMP_SR register.

Bits 15:0 Reserved, must be kept at reset value.

29.8.3 Comparator configuration register (COMP_CFGR)

The COMP_CFGR is the COMP configuration register.

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	BLANKING[3:0]				Res.	Res.	INPSEL[1:0]		INMSEL[3:0]			
rw				rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	PWRMODE[1:0]		DGLT	HYST[2:0]			Res.	ITEN	Res.	WINOUT	POLARITY	SCALEN	BRGEN	EN
		rw	rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bit 31 **LOCK:** Lock bit

This bit is set by software and cleared by a hardware system reset. It locks the whole content of the COMP configuration register COMP_CFGR[31:0] and COMP_FLTR register.
0: COMP_CFGR, COMP_FLTR register is read/write
1: COMP_CFGR, COMP_FLTR and COMP_OR registers are read-only

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:24 **BLANKING[3:0]:** COMP channel blanking source selection bits

Bits of this field are set and cleared by software (only if LOCK is not set).
The field selects the input source for COMP channel output blanking, see *Table 285: COMP blanking internal input signals (COMP1, COMP2, COMP3, COMP4)* and *Table 286: COMP blanking internal input signals (COMP5, COMP6, COMP7, COMP8)*.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **INPSEL[1:0]**: COMP channel non-inverting input selection bit

This bit is set and cleared by software (only if LOCK is not set), see *Table 281: COMP_INP non-inverting input signals (COMP1, COMP2, COMP3, COMP4)* and *Table 282: COMP_INP non-inverting input signals (COMP5, COMP6, COMP7, COMP8)*.

Bits 19:16 **INMSEL[3:0]**: COMP channel inverting input selection field

These bits are set and cleared by software (only if LOCK is not set). They select which input is connected to the minus input of COMP channel, see *Table 283: COMP_INM inverting input signals (COMP1, COMP2, COMP3, COMP4)* and *Table 284: COMP_INM inverting input signals (COMP5, COMP6, COMP7, COMP8)*.

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

Bits 13:12 **PWRMODE[1:0]**: COMP power mode selector

This bitfield is controlled by software (only if LOCK is not set). It selects the power consumption and, as a result, the speed of the comparator (refer to product datasheet).

Bit 11 **DGLT**: DEGLITCHER enable

This bit is controlled by software (only if LOCK not set). It enables the operation of the deglicher.
0: Disable
1: Enable

Bits 10:8 **HYST[2:0]:** COMP channel hysteresis selection bits

These bits are set and cleared by software (only if LOCK is not set). They select the hysteresis voltage of the COMP channel (refer to product datasheet).

Bit 7 Reserved, must be kept at reset value.

Bit 6 **ITEN:** COMP channel interrupt enable

This bit is set and cleared by software (only if LOCK is not set). This bit enables the interrupt generation of the COMP channel.

0: Interrupt generation disabled for COMP channel

1: Interrupt generation enabled for COMP channel

Bit 5 Reserved, must be kept at reset value.

Bit 4 **WINOUT**: COMP output selector

This bit is controlled by the software (only if LOCK is not set). It selects the comparator window mode output.

0: comp_out = comp_value

1: comp_out = winout (window mode output)

Bit 3 **POLARITY:** COMP polarity selection bit

This bit is set and cleared by software (only if LOCK is not set). It inverts COMP polarity.

0: COMP output is not inverted

1: COMP output is inverted

Bit 2 **SCALEN**: Voltage scaler enable bit

This bit is set and cleared by software (only if LOCK is not set). This bit enables the comp_int_vref scaler.

0: comp_int_vref scaler disabled

1: comp_int_vref scaler enabled

Bit 1 **BRGEN:** Scaler bridge enable

This bit is set and cleared by software (only if LOCK is not set). This bit enables the bridge of the scaler.

0: comp_int_vref scaler resistor bridge disabled

1: comp_int_vref scaler resistor bridge enabled

Bit 0 **EN:** COMP channel enable bit

This bit is set and cleared by software (only if LOCK is not set). It enables the COMP channel.

0: Disable

1: Enable

29.8.4 Comparator filter configuration register (COMP_FLTR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **FSEL[3:0]:** COMP digital filter configuration selection
Bits of this field are set and cleared by software (only if LOCK is not set).They select the digital filter configuration of the COMP.
0000: Filter disabled
0001: filter_sampling = comp_pclk, N = 2
0010: filter_sampling = comp_pclk, N = 4
0011: filter_sampling = comp_pclk, N = 8
0100: filter_sampling = filter_clk/2, N = 6
0101: filter_sampling = filter_clk/2, N = 8
0110: filter_sampling = filter_clk/4, N = 6
0111: filter_sampling = filter_clk/4, N = 8
1000: filter_sampling = filter_clk/8, N = 6
1001: filter_sampling = filter_clk/8, N = 8
1010: filter_sampling = filter_clk/16, N = 5
1011: filter_sampling = filter_clk/16, N = 6
1100: filter_sampling = filter_clk/16, N = 8
1101: filter_sampling = filter_clk/32, N = 5
1110: filter_sampling = filter_clk/32, N = 6
1111: filter_sampling = filter_clk/32, N = 8

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

Bits 1:0 **FPSC[1:0]:** COMP digital filter configuration selection
Bits of this field are set and cleared by software (only if LOCK is not set). They select the digital filter clock configuration of the COMP:
00: filter_clk = comp_pclk
01: filter_clk = comp_pclk/2
10: filter_clk = comp_pclk/4
11: filter_clk = comp_pclk/8

29.8.5 Comparator option register (COMP_OR)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bits 31:0 Reserved, must be kept at reset value.

29.8.6 COMP register map

The following table summarizes the comparator registers.

Table 290. COMP register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**COMP_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CIF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CVAL
	Reset value																0																0
0x04	**COMP_ICFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCIF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																0																
0x08	**COMP_CFGR**	LOCK	Res	Res.	Res.	BLANKING[3:0]				Res.	Res.	INPSEL[1:0]		INMSEL[3:0]				Res.	Res.	PWRMODE[1:0]		DGLT	HYSTO[2:0]			Res.	ITEN	Res.	WINOUT	POLARITY	SCALEN	BRGEN	EN
	Reset value	0				0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0		0		0	0	0	0	0
0x0C	**COMP_FLTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	FPSC[2:0]	
	Reset value																					0	0	0	0							0	0
0x10	**COMP_OR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

30 Audio digital filter (ADF)

30.1 ADF introduction

The audio digital filter (ADF) is a high-performance module dedicated to the connection of external sigma-delta (∑Δ) modulators. It is mainly targeted for the following applications:

- audio capture signals
- metering

The ADF features one digital serial interface (SITF0) and one digital filter (DFLT0) with flexible digital processing options in order to offer up to 24-bit final resolution.

The ADF serial interface supports several standards allowing the connection of various ∑Δ modulator sensors:

- SPI interface
- Manchester coded 1-wire interface
- PDM interface

The ADF converts an input data stream into clean decimated digital data words. This conversion is done thanks to low-pass digital filters and decimation blocks. In addition, it is possible to insert a high-pass filter.

The conversion speed and resolution are adjustable according to configurable parameters for digital processing: filter bypass, filter order, decimation ratio. The maximum output data resolution is up to 24 bits. There are two conversion modes: single conversion and continuous modes. The data can be automatically stored in a system RAM buffer through DMA, thus reducing the software overhead.

A sound activity detector (SAD) is available for the detection of sounds or voice signals. The SAD is connected at the output of the DFLT0. Several parameters can be programmed in order to adjust properly the SAD to the sound environment. The SAD strongly reduces the power consumption by preventing the storage of samples into the system memory, as long as the observed signal does not match the programmed criteria.

The digital processing is performed using only the kernel clock. The ADF requests the bus interface clock (AHB clock) only when data must be transfered or when a specific event requests the attention of the system processor.

30.2 ADF main features

- AHB Interface
- 1 serial digital input:
  - configurable SPI interface to connect various digital sensors
  - configurable Manchester coded interface support
  - compatible with PDM interface to support digital microphones
- 2 common clocks input/output for ∑Δ modulators
- 1 flexible digital filter path including:
  - A MCIC filter configurable in $Sinc^4$ or $Sinc^5$ filter with an adjustable decimation ratio
  - A reshape filter to improve the out-off band rejection and in-band ripple

  - A high-pass filter to cancel the DC offset
  - Gain control
  - Saturation blocks
- Clock absence detector
- Sound activity detector
- 24-bit signed output data resolution
- Continuous or single conversion
- Possibility to delay the selected bitstream
- One trigger input
- Autonomous functionality in Stop mode(s)
- DMA can be used to read the conversion data
- Interrupts services

30.3 ADF implementation

Table 291. ADF features(1)

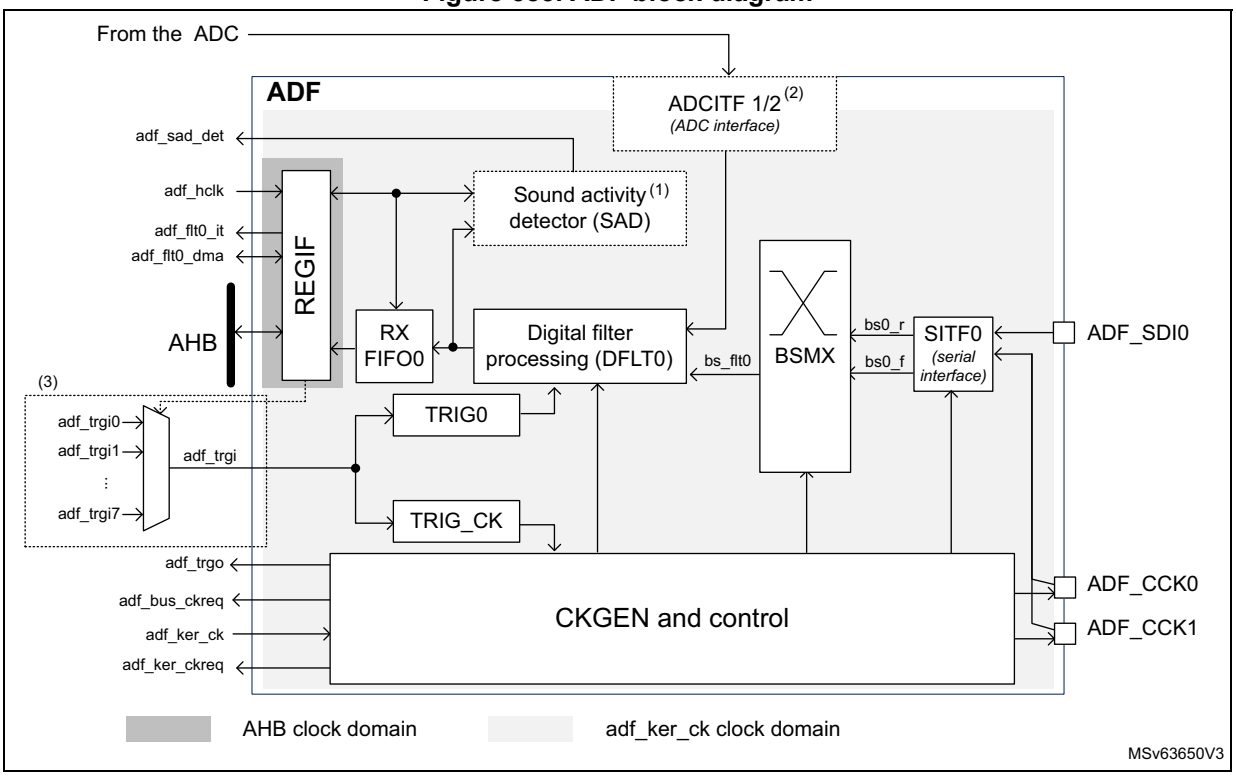
Mode or feature	ADF1
Number of filters (DFLTx) and serial interfaces (SITFx)	1
MDF_CKIy/ADF_CKI0 connected to pins	-
Sound activity detection (SAD)	X
RXFIFO depth (number of 24-bit words)	4
ADC connected to ADCITF1	ADC1
ADC connected to ADCITF2	ADC2
Motor dedicated features (SCD, OLD, OEC, INT, snapshot, break)	-
Main path with CIC4, CIC5	X
Main path with CIC1,2, 3 or FastSinc	-
RSFLT, HPF, SAT, SCALE, DLY, Discard functions	X
Autonomous in Stop modes	X(2)

1. ‘X’ = supported, ‘-’ = not supported.
2. Only Stop 0 and Stop 1 modes.

30.4 ADF functional description

30.4.1 ADF block diagram

Figure 385. ADF block diagram



1. Refer to *Section 30.3: ADF implementation* to check if the SAD is available.
2. Refer to *Section 30.3: ADF implementation* to check if the ADCITF is available, and which ADCs are connected.
3. The number of trigger inputs depends on the product. Refer to *Table 294* for details.

30.4.2 ADF pins and internal signals

Table 292. ADF external pins

Pin name	Pin type	Remarks
ADF_SDI0	Input	Data signal from external sensors
ADF_CCKy (y = 0,1)	Input/output	Clock outputs for external sensor, or common clock input from external sensors

Table 293. ADF internal signals

Signal name	Signal type	Remarks
adf_trgi	Input	Trigger inputs to control the acquisition (see *Table 294: ADF trigger connections* for details)
adf_trgo	Output	Trigger output for synchronizing with other MDF instances

Table 293. ADF internal signals (continued)

Signal name	Signal type	Remarks
adf_flt0_dma	Input/output	DMA request/acknowledge signals for the ADF processing chain
adf_flt0_it	Output	Global interrupt signals
adf_bus_ckreq	Output	Bus interface clock request output
adf_ker_ckreq	Output	Kernel clock request output
adf_ker_ck	Input	Kernel clock input
adf_hclk	Input	AHB bus interface clock input
adf_sad_det	Output	SAD sound detection: 1 means that detecting sound
adf_adcitf1_dat[15:0]	Input	ADCITF1 data input
adf_adcitf2_dat[15:0]	Input	ADCITF2 data input

Table 294. ADF trigger connections

Trigger name	Direction	Trigger source/destination
adf_trgi0	Input	exti15
adf_sad_det	Output	hsp1_trg_in19

Table 295. MDF ADC data connections

ADC data bus name	ADC source
mdf_adcitf1_dat[15:0]	adc1_dat
mdf_adcitf2_dat[15:0]	adc2_dat

30.4.3 Serial input interface (SITF)

The SITF0 input interface allows the connection of the external sensor to the digital filter via the bitstream matrix (BSMX). The SITF0 can be configured in the following modes:

- LF_MASTER SPI mode (low-frequency)
- normal SPI mode
- Manchester mode

The data from the serial interface is routed to the filter in order to perform the PDM to PCM conversion and the sound activity detection.

The serial interface is enabled by setting the SITFEN bit to 1. Once the interface is enabled, it receives serial data from the external ∑Δ modulator.

Note: Before enabling the *serial interface, the user must insure that the adf_proc_ck is already enabled (see Section 30.4.5: Clock generator (CKGEN) for details).*

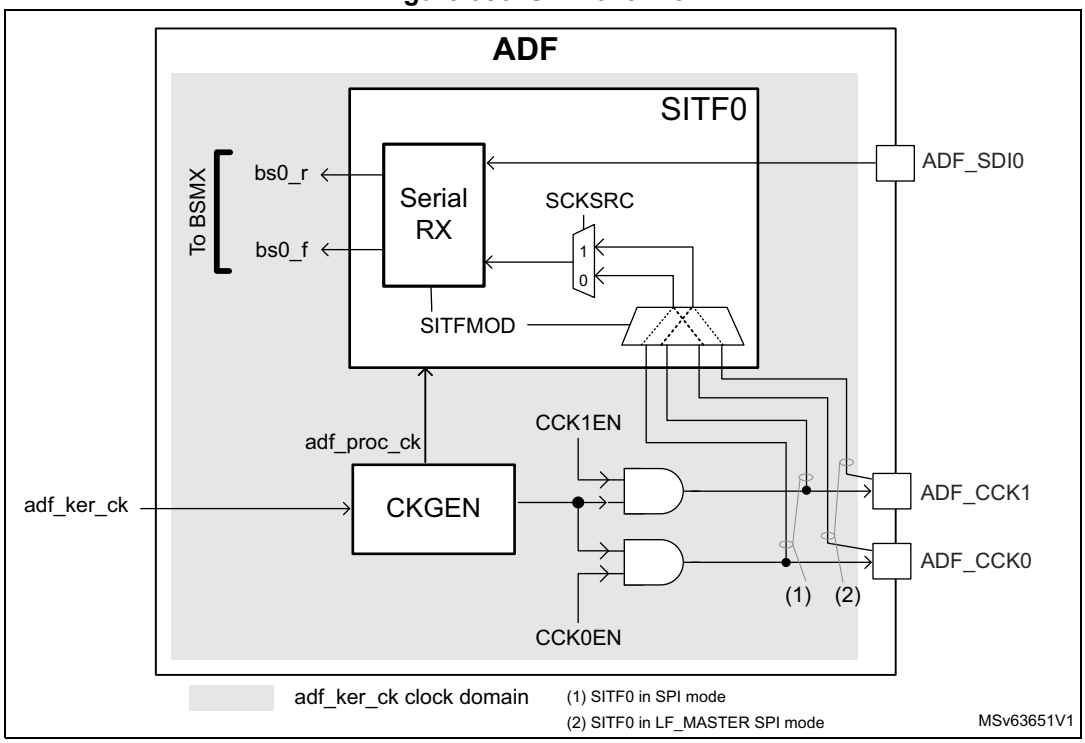
The SITF0 is controlled via the *ADF serial interface control register 0 (ADF_SITF0CR)*.

As shown in the *Figure 386*, ADF_CCK0 or ADF_CCK1 can be selected as clock source, in order to sample the incoming bitstream:

- If the serial interface is programmed in SPI mode, the selected clock source is a copy of the clock present on the ADF_CCK0 or ADF_CCK1 pin.
- If the serial interface is programmed in LF_MASTER SPI mode, the selected clock source is the clock directly provided by the CCKDIV to the ADF_CCK0 or ADF_CCK1 pin.

See *Table 296* for additional information.

Figure 386. SITF overview



LF_MASTER and normal SPI modes

The LF_MASTER SPI mode is a special mode allowing the use of an adf_proc_ck clock frequency, only two times bigger than the sensor clock. This mode is dedicated to low-power use-cases, using low-speed sensors.

In LF_MASTER SPI mode, the ADF must provide the bitstream clock to the external sensors via ADF_CCK0 and ADF_CCK1 pins. The ADF receives the bitstream data via the serial data input ADF_SDI0.

For the SITF0, the application must select the same clock than the one provided to the external sensor (ADF_CCK0 or ADF_CCK1), in order to guarantee optimal timing performances. This selection is done via SCKSRC[1:0].

The normal SPI interface is a more flexible interface than the LF_MASTER SPI, but the adf_proc_ck frequency must be at least four times higher than the sensor clock.

The application can select ADF_CCK0 or ADF_CCK1 clock for the capture of the data received via the ADF_SDI0 pin.

The ADF can generate a clock to the sensors via ADF_CCK0 or ADF_CCK1 if needed.

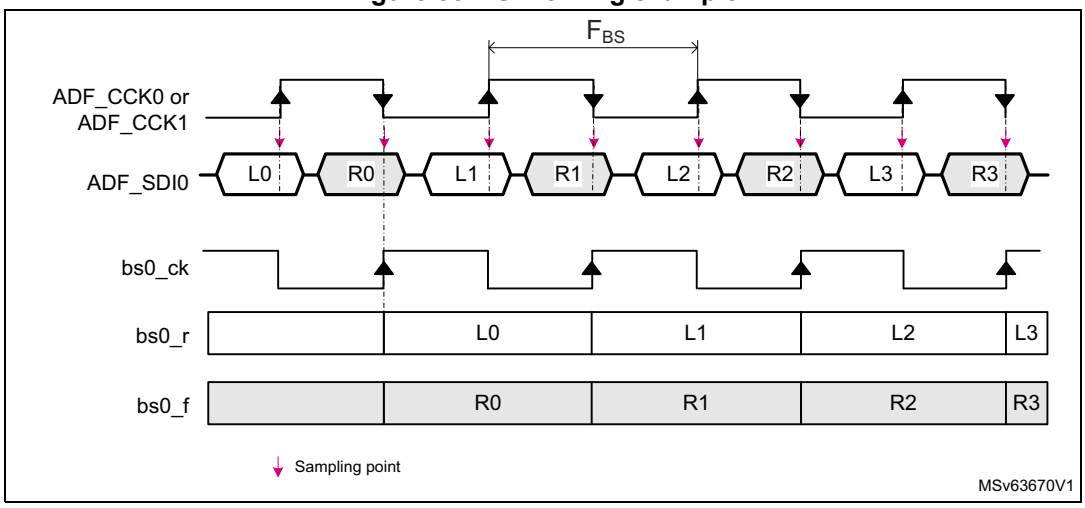
For all SPI modes, the serial data is captured using the rising and the falling edge of the selected clock. The SITF0 always provides the following bitstreams:

- bitstream received using the bitstream clock falling edge (bs0_f)
- bitstream received using the bitstream clock rising edge (bs0_r)

According to the sensors connected, one of the two bitstreams may not be available.

The application can select the wanted stream via the BSMX matrix.

Figure 387. SPI timing example



To properly synchronize/receive the data stream, the adf_proc_ck frequency must be adjusted according to the constraints listed in *Table 297*.

Clock absence detection

A no-clock-transition period may be detected when the serial interface works in normal SPI mode. This feature can be used to detect a clock failure in the SPI link.

The application can program a timeout value via the STH[4:0] bitfield of the SITF0. If the ADF does not detect clock transitions for a duration of STH[4:0] x $T_{adf_proc_ck}$, then the CKABF flag is set.

An interrupt can be generated if CKABIE is set to 1. The STH[4:0] bitfield is in the *ADF serial interface control register 0 (ADF_SITF0CR)*.

When the serial interface is enabled, the CKABF flag remains to 1 until a first clock transition is detected.

To avoid spurious clock absence detection, the following sequence must be respected:

1. Configure the serial interface in normal SPI mode and enable it.
2. Clear the CKABF flag by writing CKABF bit to 1.

   If no clock transition is detected on the serial interface, the hardware immediately sets the CKABF flag to 1.

3. Read the CKABF flag:
   – If CKABF = 1, go back to step 2.
   – If CKABF = 0, a clock has been detected. The CKABIE bit can be set to 1 if the application wants an interrupt on detection of a clock absence.

Note: The clock absence detection feature is not available in the LF_MASTER SPI mode.

Manchester mode

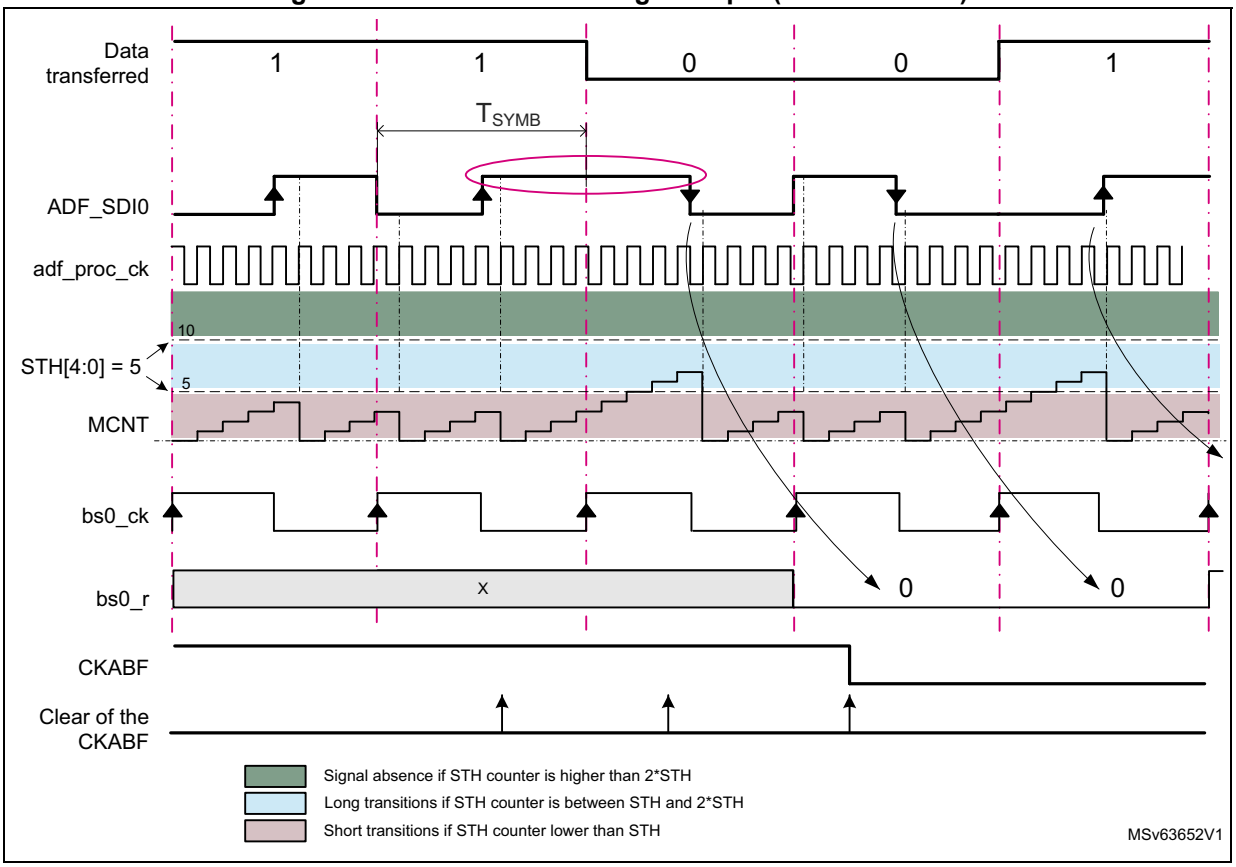
In Manchester coded format, the ADF receives data stream from the external sensor via the ADF_SDI0 pin only.

The ADF_CCK0 and ADF_CCK1 pins are not needed in this mode.

Decoded data and clock signals are recovered from serial stream after Manchester decoding. They are available on bs0_r. There are two possible settings of Manchester codings:

- signal rising edge decoded as 0 and signal falling edge decoded as 1
- signal rising edge decoded as 1 and signal falling edge decoded as 0

Figure 388. Manchester timing example (SITFMOD = 11)



To decode the incoming Manchester stream, the user must program STH[4:0] in the *ADF serial interface control register 0 (ADF_SITF0CR)*. The STH[4:0] bitfield is used by the SITF0 to estimate the Manchester symbol length and to detect a clock absence. An internal counter (MCNT) is restarted every time a transition is detected in the ADF_SDI0 input. It is used to detect short transitions, long transitions or clock absence. A long transition indicates

that the data value changed. *Figure 388* shows a case where the OVR is around height and STH[4:0] = 5.

The estimated Manchester symbol rate (T_{SYMB}) must respect the following formula:

$$(STH + 1) \times T_{adf_proc_ck} < T_{SYMB} < (2 \times STH \times T_{adf_proc_ck})$$

It is recommended to compute STH as follows:

$$STH[4:0] = round\left(\frac{(2 \times OVR) - 1}{3}\right)$$

where OVR represents the ratio between the adf_proc_ck frequency and the expected Manchester symbol frequency. OVR must be higher than five, and the adf_proc_ck clock must be adjusted according to the constraints listed in *Table 297*.

The clock absence flag CKABF is set to 1 when no transition is detected during more than 2 x STH[4:0] x $T_{adf_proc_ck}$, or when the SITF0 is not yet synchronized to the incoming Manchester stream. In addition, an interrupt can be generated if the bit CKABIE is set to 1.

When the serial interface is enabled, the ADF must first be synchronized to the incoming Manchester stream. The synchronization ends when a data transition from 0 to 1 or from 1 to 0 (pink circle in the *Figure 388*) is detected.

The end of the synchronization phase can be checked by following the software sequence:

1. Clear the CKABF flag in the *ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)* by writing CKABF bit to 1. If the serial interface is not yet synchronized the hardware immediately set the CKABF flag to 1.
2. Read the CKABF flag:
   – If CKABF= 1, go back to step 1.
   – If CKABF = 0, the Manchester interface is synchronized and provides valid data.

Programming example

In the following example, the ADF kernel clock frequency ($F_{adf_ker_ck}$) is 100 MHz and the received Manchester stream is at about 6 MHz (F_{SYMB}):

1. Provide a valid adf_proc_ck to the SITF0.

   The adf_proc_ck frequency must be at least six times higher than the Manchester symbol frequency (means at least 36 MHz).

   PROCDIV is programmed to 1 to perform a division by two of the kernel clock. In that case, $F_{adf_proc_ck}$ = 50 MHz (8.33 times higher than the Manchester symbol frequency).
2. Compute STH.

   OVR is given by: OVR = $F_{adf_proc_ck}$ / F_{SYMB} = 50 MHz / 6 MHz = 8.33.

   Then $STH[4:0] = round\left(\frac{(2 \times 8.33) - 1}{3}\right) = 5$

   The minimum allowed frequency for the Manchester stream is then:

   1 / (2 × STH × $T_{adf_proc_ck}$) = 1 / (10 x 20 ns) = 5 MHz

   The maximum allowed frequency for the Manchester stream is then:

   1 / ((STH+1) × $T_{adf_proc_ck}$) = 1 / (6 x 20 ns) = 8.33 MHz

30.4.4 ADC slave interface (ADCITF)

The ADCs are not always connected to the ADF. Refer to *Section 30.3* to check the situation for this product.

The ADF allows the connection of up to two ADCs to the filter path. For the filter, the DATSRC[1:0] bitfield in the *ADF digital filer configuration register 0 (ADF_DFLT0CICR)* allows the application to select data from the ADCs.

Warning: The ADF does not support receiving interleaved data from one of the ADCITF input.

30.4.5 Clock generator (CKGEN)

The RCC (reset and clock controller) provides the following clocks to the ADF:

- AHB clock (adf_hclk) used for the register interface
- kernel clock (adf_ker_ck) mainly used by all other parts of the circuit via the CKGEN

These clocks are not supposed to be phase locked, so all signals crossing those clock domains are resynchronized.

The clock generator (CKGEN) is responsible of the generation of the processing clock, and the clock provided to the ADF_CCK0 and ADF_CCK1 pins. All those clocks are generated from the adf_ker_ck.

The processing clock (adf_proc_ck) is used to run all the signal processing and to re-sample the incoming serial or parallel stream.

To adapt the kernel clock frequency provided by the RCC, the following dividers are available:

- PROCDIV[6:0] used to adapt the kernel clock frequency to the constraints of the parallel and serial interfaces, and to the processing blocks
- CCKDIV[3:0] used to adapt the frequency of the ADF_CCK0 and ADF_CCK1 clocks

PROCDIV[6:0] and CCKDIV[3:0] must be programmed when no clock is provided to the dividers (CKGDEN = 0).

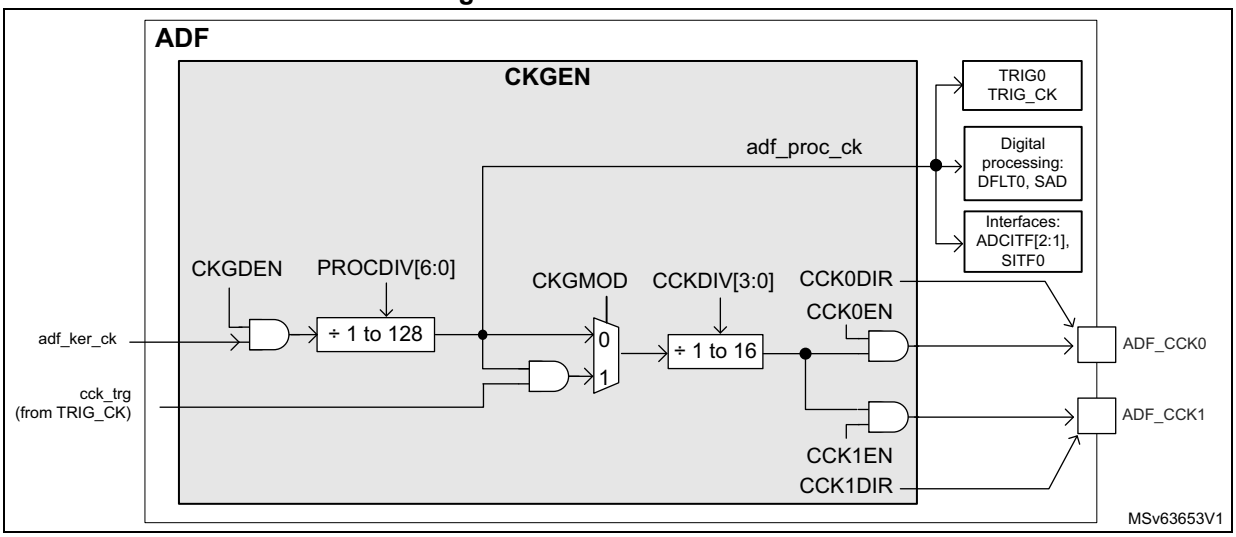
The adf_proc_ck generation is controlled by CKGDEN.

In addition, the CKGMOD bit allows the application to define the way to trigger the CCKDIV divider:

- When CKGMOD = 0, the CCKDIV divider is started as soon as CKGDEN is set to 1.
- When CKGMOD = 1, the CCKDIV divider is started when CKGDEN is set to 1 and the programmed trigger condition occurred.

All the bits and fields controlling the CKGEN are in the *ADF clock generator control register (ADF_CKGCR)*.

Figure 389. CKGEN overview



The trigger logic for CKGEN is handled by the block TRG_CK. As shown in *Figure 394*, the CCKDIV divider can be triggered on the rising or falling edge of an external trigger source. When the proper trigger condition occurs, the cck_trg signal goes to high, allowing the CCKDIV divider to start. The TRG_CK logic is reset when CKGDEN is set to 0.

This feature can be helpful to synchronize the ADF_CCKy (y = 0,1) clock of several ADF instances, or to synchronize the clock generation to a timer event.

The application can control the activation of the ADF_CCK0 or ADF_CCK1 pin thanks to CCK0EN/CCK1EN and CCK0DIR/CCK1DIR bits:

- CCKyEN is used to enable the CCKDIV, and thus generates a clock for the external sensors.
- CCKyDIR is used to control the direction of the ADF_CCKy pin (input or output)

Table 296. Control of the common clock generation(1)

CCKyEN	CCKyDIR	Description
0	0	The ADF_CCKy pin is in input. An external clock can be connected to the ADF_CCKy pin and used by the SITF0 in order to decode the serial stream
0	1	The ADF_CCKy pin is in output. No clock is generated. The ADF_CCKy pin is driven low.
1	1	The ADF_CCKy pin is in output. A clock is generated on the ADF_CCKy pin. The SITF0 can use this pin as clock source in order to decode the serial stream

1. The configuration with CCKyEN = 1 and CCyDIR = 0 is not shown must be avoided (no interest).

Note: The adf_proc_ck must be enabled (by CKGDEN = 1) before enabling other blocks (such as SITF0 or DFLT0).

CKGEN activation sequence example

1. Set CKGDEN to 0.
2. Wait for CKGACTIVE = 0. If CKGDEN was previously enabled, this phase can take two periods of adf_hclk, and two periods of adf_proc_ck.
3. Program PROCDIV[6:0], CKGMOD, CCKDIV[3:0], TRGSRC[3:0], TRGSENS, CCK1EN and CCK0EN.
4. Set CKGDEN to 1.

When needed, at any moment, CCK[1:0]EN bitfield value can be changed without disabling the clock generator.

Clock frequency constraints

Table 297 shows the frequency constraints to receive and process properly the samples.

Note: The reshape filter (RSFLT) needs up to 24 cycles of adf_proc_ck clock to process one sample.

Table 297. Clock constraints with respect to the incoming stream(1)

SITF0 mode	ADF clock constraints	
	With RSFLT disabled	With RSFLT enabled
LF_MASTER SPI	F_{ADF_CCKy} max frequency limited to 5 MHz	
	$F_{adf_proc_ck} > 2 \times F_{ADF_CCKy}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$	$F_{adf_proc_ck} > 24 \times F_{ADF_CCKy} / (MCICD+1)$ and $F_{adf_proc_ck} > 2 \times F_{ADF_CCKy}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$
MASTER SPI SLAVE SPI	F_{ADF_CKx} max frequency limited to 25 MHz	
	$F_{adf_proc_ck} > 4 \times F_{ADF_CCKy}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$	$F_{adf_proc_ck} > 24 \times F_{ADF_CCKy} / (MCICD+1)$ and $F_{adf_proc_ck} > 4 \times F_{ADF_CCKy}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$
Manchester	F_{SYMB} max frequency limited to 20 MHz	
	$F_{adf_proc_ck} > 6 \times F_{SYMB}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$	$F_{adf_proc_ck} > 24 \times F_{ADF_CCKy} / (MCICD+1)$ and $F_{adf_proc_ck} > 6 \times F_{SYMB}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$

1. F_{ADF_CCKy} represents the frequency of clock received via ADF_CCKy, or generated via ADF_CCKy. F_{SYMB} represents the frequency of the received symbol rate for Manchester mode.

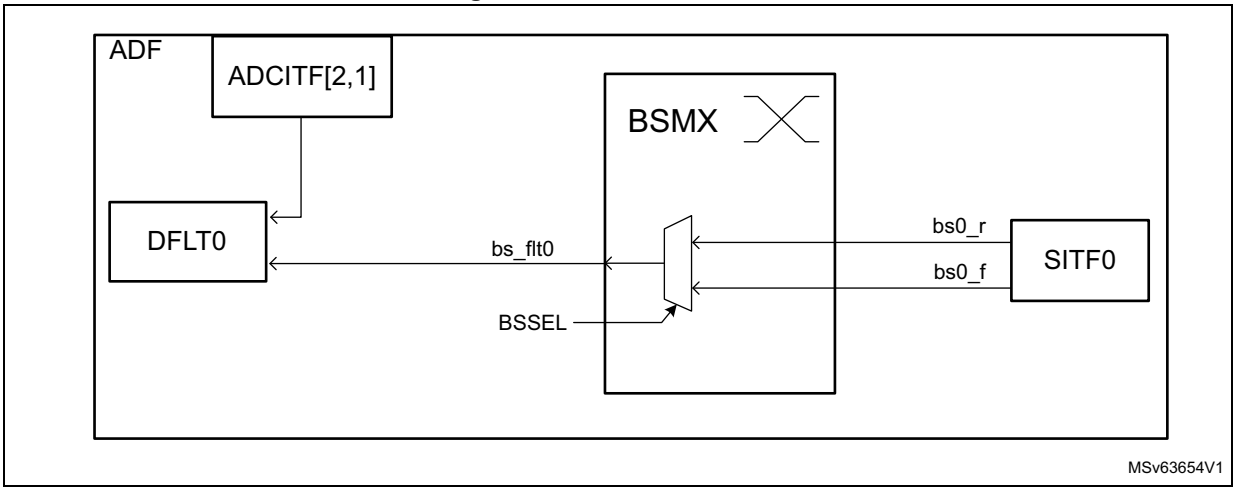
30.4.6 Bitstream matrix (BSMX)

The BSMX receives the bitstreams from the serial interface SITF0 and provides the selected stream to the digital filter DFLT0.

As shown in *Figure 385*, the SITF0 provides two bitstreams (bs0_r and bs0_f) to the BSMX.

The application to select the wanted stream via the *ADF bitstream matrix control register 0 (ADF_BSMX0CR)*. This selection is intended to be static.

Figure 390. BSMX overview



BSMX programming sequence example

The BSSEL[4:0] bitfield cannot be changed if the DFLT0 is enabled. The following steps are needed to change the value of BSMX:

1. Set DFLTEN of DFLT0 to 0.
2. Wait for BSMXACTIVE = 0.
3. Program BSSEL[4:0].
4. Set DFLTEN of DFLT0 to 1.

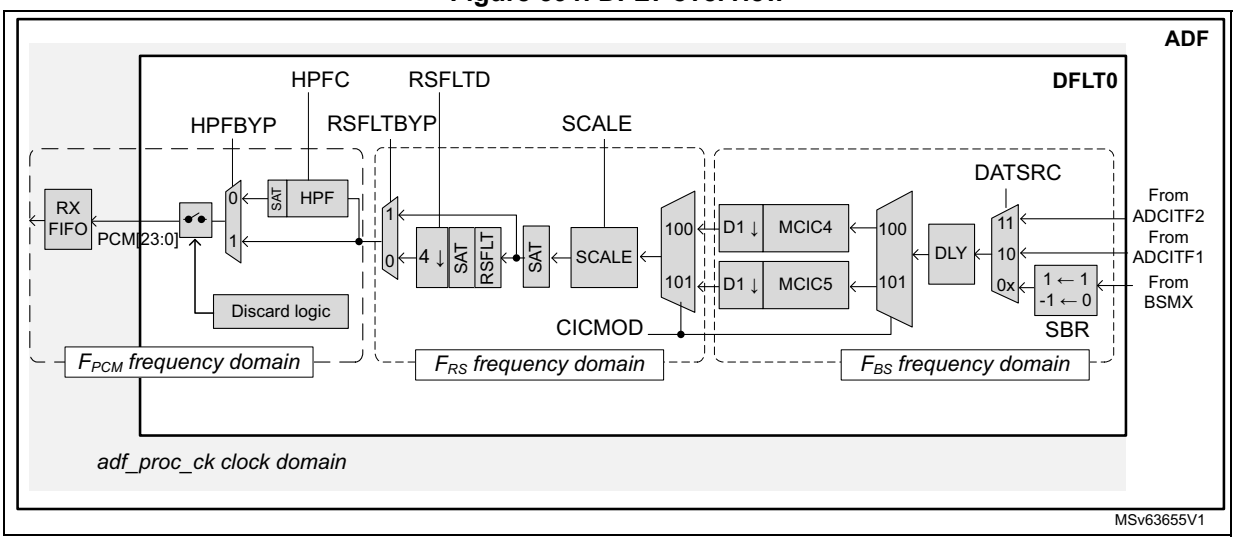
30.4.7 Digital filter processing (DFLT)

The digital filter processing includes the following sub-blocks:

- symbol remap (SBR)
- clock skipper delay (DLY)
- MCIC decimation filter that can be configured in $Sinc^4$ or $Sinc^5$
- gain control (SCALE)
- signal saturation (SAT)
- reshape filter (RSFLT)
- high-pass filter (HPF)
- receive RXFIFO

Figure 391 shows the filter path configuration according to CICMOD[2:0]. Several configuration bits are available to configure the digital filter to the application needs.

Figure 391. DFLT overview



Symbol remap and source selection

The symbol remap (SBR) converts the bitstream selected by the BSMX into data usable by the filter path. More especially:

- The high levels are converted into a 16-bit signed number + 1.
- The low levels are converted into a 16-bit signed number - 1.

The signal source of the digital filter can be selected via DATSRC[1:0] between the two following:

- data coming from the BSMX
- data coming from one of the ADC interfaces (ADCITF2 or 1)

Programmable micro-delay control (DLY)

The digital filter has a delay line that allows the timing adjustment of each stream with the resolution of the bitstream clock.

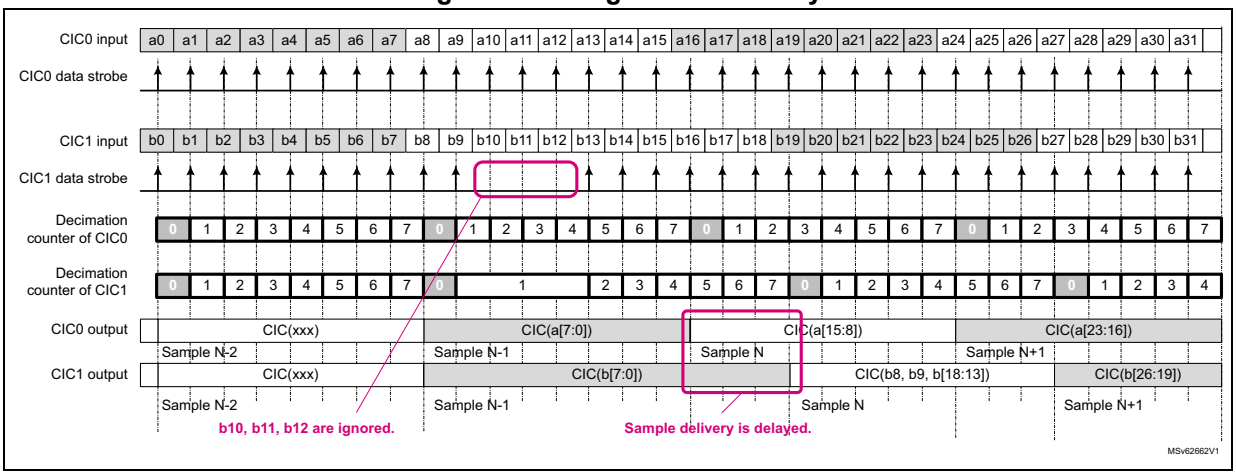
This feature is particularly helpful in the case of microphone beam forming applications where delays smaller than the final sampling rate must be applied to the incoming stream. This feature can be used when the ADF is synchronized with another MDF instance (if present in the product) for a beam forming application for example.

The delay is performed by discarding a given number of samples from the selected input stream, before samples enter into the CIC filter. This data discarding is done by skipping a given number of data strobe, preventing the CIC filter to take into account those data.

When the wanted amount of data strobe has been skipped, the next incoming samples are strobed normally.

Table 392 shows an example on how to apply dynamically small delay to an incoming stream. For simplification, the CIC filter performs a decimation by height in this example. CIC1 represents the CIC included in the ADF and CIC0 represents a filter from another MDF instance (if present in the product).

Figure 392. Programmable delay



The CIC of the ADF (CIC1) receives a command in order to skip three incoming samples. So the input samples named b10, b11 and b12 are not processed by CIC1. As a consequence, the output sample N+1 generated by CIC0 is built from input samples a[23:16] while the sample N+1 of CIC1 is built from input samples b[26:19].

Finally, the non-skipped data stream looks delayed by three bitstream periods.

Note: When the input data strobes are skipped, the decimation counter remains frozen. As a consequence, the samples delivered by the CIC1 are a bit delayed.

The following steps are needed to program the amount of bitstream clock periods to be skipped:

1. Wait for SKPBF equal to 0.
2. Write SKPDLY[6:0] to the wanted number of bitstream clock periods to be skipped. The SKPBF flag goes immediately to 1, indicating that the delay value entered into SKPDLY[6:0] is under process.
   - If the DFLT0 is not yet enabled (DFLTEN = 0), then the DLY logic waits for DFLTEN = 1. When the application sets DFLTEN to 1, the DLY logic starts to skip the amount of wanted data strobes.
   - If the DFLT0 is already enabled (DFLTEN = 1), then the DLY logic immediately starts to skip the amount of wanted data strobes.

   When the ADF skipped the amount of wanted data strobes, then SKPBF goes back to 0.
3. If the application needs to skip more data strobes, then the operation must be restarted from step 1.

The effect of the delay performed with this mechanism is cumulative as long as the ADF is enabled. If the application performs a D1 delay followed by a D2 delay, then all other active filters are delayed by D1 + D2.

Note: If SKPDLY[6:0] is written when SKPBF = 1, the write operation is ignored.

Cascaded-integrator-comb (CIC) filter

The CIC digital filters are an efficient implementation of low-pass filters, often used for decimation and interpolation. The CIC frequency response is equal to a $Sinc^N$ function, this is why they are often called Sinc filters.

The Sinc^N digital filter embedded into the ADF can be configurable in Sinc^4 or Sinc^5, according to CICMOD:

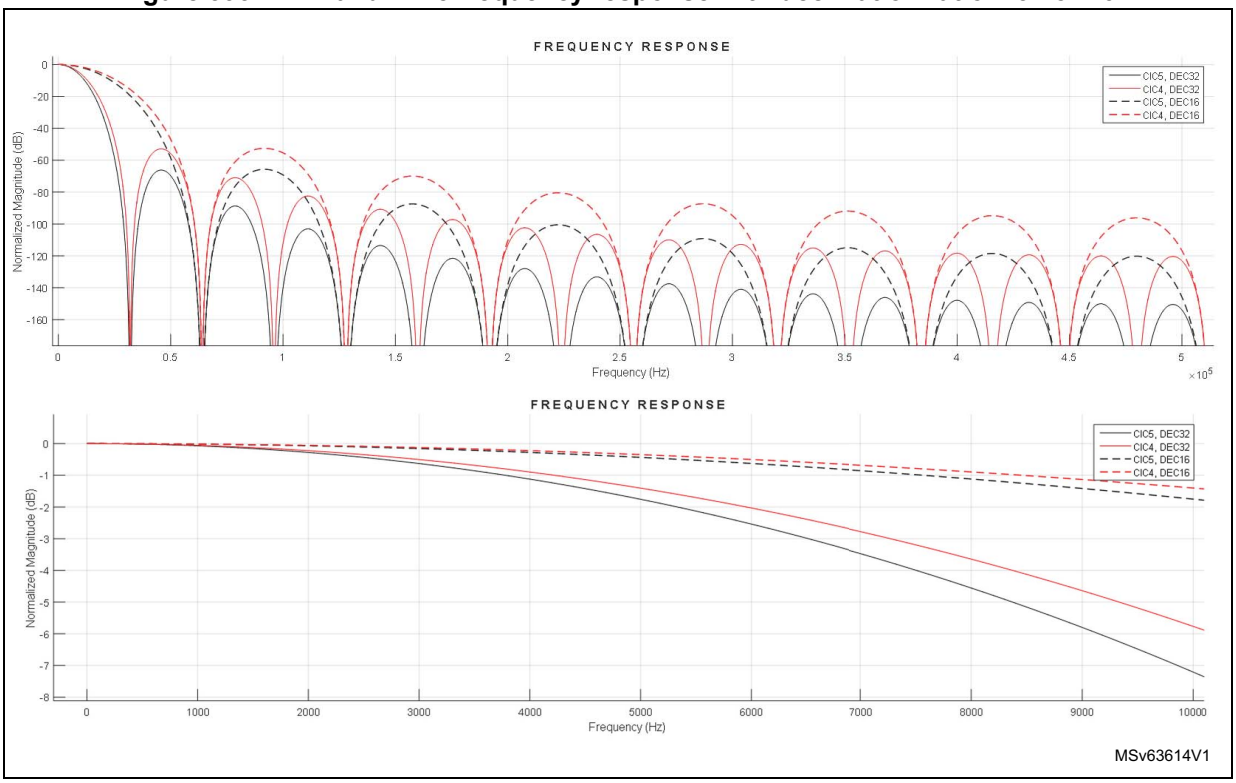
- If CICMOD[2:0] = 4, Sinc^4 is selected.
- If CICMOD[2:0] = 5, Sinc^5 is selected.

The filters have the following transfer function:

$$H(z) = \left(\frac{1 - z^{-D}}{1 - z^{-1}}\right)^N$$

where N can be 4 or 5, and D is the decimation ratio. D is equal to MCICD+1.

Figure 393. CIC4 and CIC5 frequency response with decimation ratio = 32 or 16



CIC output data size

The size of samples delivered by the CIC (DS_{CIC}), depends on the following parameters:

- CIC order (N)
- CIC decimation ratio (D)
- data size of the input stream (DSIN)

The CIC order and decimation ratio must be programmed in order to insure that the data size does not exceed the 26-bit CIC capability.

The following formula gives the output data size (DS_{CIC}) according to the parameters above.

$$DS_{CIC} = \left(\frac{N \times Ln(D)}{Ln(2)}\right) + DS_{IN}$$

and the CIC gain is given by this formula:

$$G_{CIC} = (D)^N$$

The decimation ratio can be adjusted from 2 to 512 for the CIC filter.

Table 298 gives some data output size in bits for some decimation values, when the data source is a full-scale signal coming from the serial interface or from a 12-bit ADC.

Note: DS_{IN} = 1 bit for a serial bitstream, but can be up to 16 bits when coming from the ADCITF.

Table 298. Data size according to CIC order and CIC decimation values

Decimation	Data size (bits) when DS_{IN} = 1 bit (data from SITF0)		Data size (bits) when DS_{IN} = 12 bits (data from ADCITF)	
	Sinc^4	Sinc^5	Sinc^4	Sinc^5
4	9	11	20	22
8	13	16	24	-
10	15	18	26	-
12	16	19	-	-
16	17	21	-	-
20	19	23	-	-
24	20	24	-	-
32	21	26	-	-
48	24	-	-	-
64	25	-	-	-
76	26	-	-	-

Note: For a full-scale input signal, the decimation ratio must not exceed 76 for a Sinc^4 and 32 for a Sinc^5.

The LSB parts of the data provided by the CIC is not necessarily significant: it depends on the sensor performances and the ability of the CIC to reject the out-off band noise.

The sample size at CIC output can be adjusted thanks to the SCALE block.

Scaling (SCALE) and saturation (SAT)

The SCALE block allows the application to adjust the amplitude of the signal provided by the CIC, by steps of 3 dB (± 0.5 dB).

The signal amplitude can be decreased by up to 8 bits (- 48.2 dB), and can be increased by up to 12 bits (+ 72.2 dB).

The gain is adjusted by the SCALE[5:0] bitfield in the *ADF digital filer configuration register 0 (ADF_DFLT0CICR)*.

SCALE[5:0] can be changed even if the DFLT0 is enabled. During the gain transition, the signal provided by the filter is disturbed.

Due to internal resynchronization, there is a delay of some cycles of adf_proc_ck clock between the moment where the application writes the new gain, and the moment where the gain is effectively applied to the samples. If the application attempts to write a new gain value while the previous one is not yet applied, this new gain value is ignored. Reading back SCALE[5:0] informs the application on the current gain value.

Table 299 shows the possible gain values.

Table 299. Possible gain values

SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)
0x20	- 48.2	0x2B	- 14.5	0x06	+ 18.1	0x11	+ 51.7
0x21	- 44.6	0x2C	- 12.0	0x07	+ 21.6	0x12	+ 54.2
0x22	- 42.1	0x2D	- 8.5	0x08	+ 24.1	0x13	+ 57.7
0x23	- 38.6	0x2E	- 6.0	0x09	+ 27.6	0x14	+ 60.2
0x24	- 36.1	0x2F	- 2.5	0x0A	+ 30.1	0x15	+ 63.7
0x25	- 32.6	0x00	0.0	0x0B	+ 33.6	0x16	+ 66.2
0x26	- 30.1	0x01	+ 3.5	0x0C	+ 36.1	0x17	+ 69.7
0x27	- 26.6	0x02	+ 6.0	0x0D	+ 39.6	0x18	+ 72.2
0x28	- 24.1	0x03	+ 9.5	0x0E	+ 42.1	-	-
0x29	- 20.6	0x04	+ 12.0	0x0F	+ 45.7	-	-
0x2A	- 18.1	0x05	+ 15.6	0x10	+ 48.2	-	-

The SAT blocks avoid having a wrap-around of the binary code when the code exceeds its maximal or minimal value.

The ADF performs saturation operations at the following levels:

- after the SCALE block (performed by the SAT block): The signal is saturated at 24 bits.
- inside the RSFLT, to insure a good filter behavior
- at the output of the HPF, to insure that the output signal does not exceed 24 bits

The SATF bit informs the application that a saturation occurred either after the SCALE, inside the RSFLT or after the HPF. In addition, an interrupt can be generated if SATIE is set to 1. As soon as a saturation is detected, the SATF flag is set to 1. It is up to the application to clear this flag in order to be able to detect a new saturation.

Those bits are in the *ADF DFLT0 interrupt enable register (ADF_DFLT0IER)* and *ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)*.

Gain adjustment policy

To get the best ADF performances, it is important to properly adjust the gain value via SCALE[5:0].

A usual way to adjust the gain is to select the SCALE[5:0] value that gives a final signal amplitude as close as possible to the 24-bit full-scale, for the maximum input signal.

A way to select the optimal gain is detailed below:

1. Check that, for the expected input signal, the data size into the CIC filter does not exceed 26 bits. This can be checked using this formula:

$$\frac{LN(SIN_{pp} \cdot D^N)}{LN(2)} < 26$$

where N represents the CIC order, D the decimation ratio and SIN_{pp} the maximum peak-to-peak amplitude of the input signal.

SIN_{pp}can take:

– a maximum peak-to-peak amplitude of 2 (± 1), for samples coming from SITF0

– A maximum peak-to-peak amplitude of 4095 (+ 2047, - 2048), for samples coming from a 12-bit ADC

Example: a $Sinc^4$ can be used with a decimation ratio of 96, if the maximum input signal does not exceed ± 0.35. Indeed:

$$\frac{LN(0.7 \cdot 96^4)}{LN(2)} \sim 25.82 \text{ bits} < 26 \text{ bits}$$

2. Adjust the SCALE value.

   To select the most appropriate SCALE value, the user must check if the RSFLT is used or not. If the RSFLT is used, the data size at SCALE output must not exceed 22 bits, otherwise the data size can be up to 24 bits.

   The SCALE value in dB must be selected using this formula:

$$SCALE_{dB} < 20 \cdot \log 10\left(\frac{2^{NB}}{SIN_{pp} \cdot D^{N}}\right)$$

   where NB is equal to 22 if RSFLT is enabled, or 24 if RSFLT is bypassed. $SCALE_{dB}$ represents the gain value selected by SCALE[5:0].

   Example: For a $Sinc^4$ with a decimation ratio of 96 and a SIN_{pp} of 0.7.

   – If the RSFLT is bypassed:

$$20 \cdot \log 10\left(\frac{2^{24}}{0.7 \cdot 96^{4}}\right) \sim -11 \text{ dB}$$

     $SCALE_{dB}$ value must be lower than - 11 dB, the closest lower value is - 12dB (SCALE[5:0] = 0x2C).

   – If the RSFLT is enabled:

$$20 \cdot \log 10\left(\frac{2^{22}}{0.7 \cdot 96^{4}}\right) \sim -23 \text{ dB}$$

     $SCALE_{dB}$ value must be lower than - 23 dB, the closest lower value is - 24.1 dB (SCALE[5:0] = 0x28).

   If SCALE[5:0] is set to a higher value, then a saturation may occur. An event flag informs the user if a saturation occurred.

Table 300 proposes gain values for different filter configurations, when the data comes from the SITF0, according to the MCIC order, and the MCIC decimation ratio. This table is not exhaustive and considers a full-scale input signal (see *Section 30.7.5: Total ADF gain* for details).

Table 300. Recommended maximum gain values versus CIC decimation ratios

CIC decimation ratio	Gain settings (dB) for configuration SITF + CICx + RSFLT (+ HPF)		Gain settings (dB) for configuration SITF + CICx (+ HPF)	
	CIC5	CIC4	CIC5	CIC4
8	33.6	51.7	45.7	63.7
12	18.1	39.6	30.1	51.7
16	3.5	27.6	15.6	39.6
20	- 6.0	21.6	6.0	33.6
24	- 12.0	15.6	0	27.6
28	- 20.6	9.5	- 8.5	21.6

Table 300. Recommended maximum gain values versus CIC decimation ratios (continued)

CIC decimation ratio	Gain settings (dB) for configuration SITF + CICx + RSFLT (+ HPF)		Gain settings (dB) for configuration SITF + CICx (+ HPF)	
	CIC5	CIC4	CIC5	CIC4
32	- 26.6	3.5	- 4.5	15.6
48	-	- 8.5	-	3.5
64	-	- 20.6	-	- 8.5

Reshaping filter (RSFLT)

In addition to the CIC, the ADF offers a reshaping IIR filter mainly dedicated to the audio application, but also usable in other applications.

When the RSFLT is used, the sample size at its input must not exceed 22 bits.

The samples at the RSFLT output can be decimated by four or not according to the RSFLTD bit in the *ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)*.

The RSFLT can be bypassed by setting RSFBYP to 1 in the *ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)*.

Table 301 shows which sampling rate must be provided to the RSFLT in order to process the most common audio streams.

The RSFLT cutoff frequency (F_C) depends on the sample rates at its input (F_{RS}), and is given by the following formula:

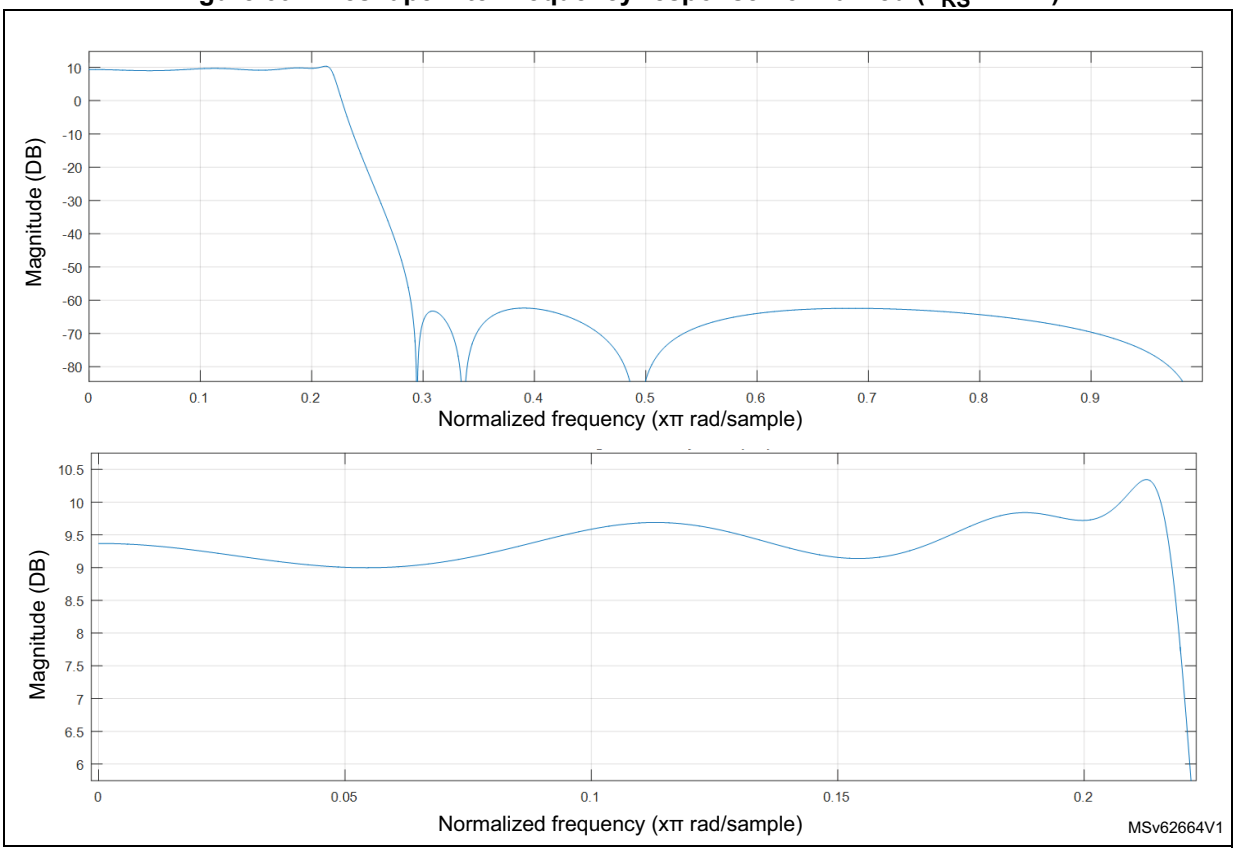
$$F_C = 0.111 \times F_{RS}$$

Table 301. Most common microphone settings

Sample rate (kHz) at RSFLT (F_{RS})	Pass band (kHz)	D2	PCM sampling rate (kHz)
32	3.55	4	8
64	7.1	4	16
128	14.2	4	32
192	21.3	4	48

Table 394 shows the frequency response of the reshape filter.

Figure 394. Reshape filter frequency response normalized (F_{RS} / 2 = 1)



The RSFLT gain is close to 9.3 dB, so the output data size is a little bit lower than 24 bits for a 22-bit wide input signal.

The RSFLT takes 24 clock cycles of adf_proc_ck clock to process one sample at F_{RS}. When the RSFLT is enabled, the application must insure that the adf_proc_ck is at least 24 times faster F_{RS}.

The RSFLT generates an event (rfovr_evt) and sets the RFOVRF flag, if the RSFLT receives a new samples while the previous one is still under processing.

When RFOVRF is set, the samples provided by the RSFLT are invalid. The application must then stop the data acquisition and provides a faster adf_proc_ck clock to the RSFLT.

High-pass filter (HPF)

The high-pass filter suppresses the low-frequency content from the final output data stream in case of continuous conversion mode. The high-pass filter can be enabled or disabled via HPFBYP in the *ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)*.

The HPF is useful when there is parasitic low-frequency noise (or DC signal) in the input data source that must be removed from the final data.

The HPF is a first order IIR filter and the cut-off frequency can be selected via HPFC[1:0] in the *ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)*, among the following values:

- 0.000625 x F_{PCM}
- 0.00125 x F_{PCM}
- 0.00250 x F_{PCM}
- 0.00950 x F_{PCM}

Table 302. HPF 3 dB cut-off frequency examples

HPFC	3 dB cut-off frequency for common F_{PCM} frequencies (Hz)		
	F_{PCM} = 8 kHz	F_{PCM} = 16 kHz	F_{PCM} = 48 kHz
0	5	10	30
1	10	20	60
2	20	40	120
3	76	152	456

The HPF output is saturated at 24 bits. The SATF flag is set if a sample is saturated.

30.4.8 Digital filter acquisition modes

The ADF offers the following modes to perform a data capture:

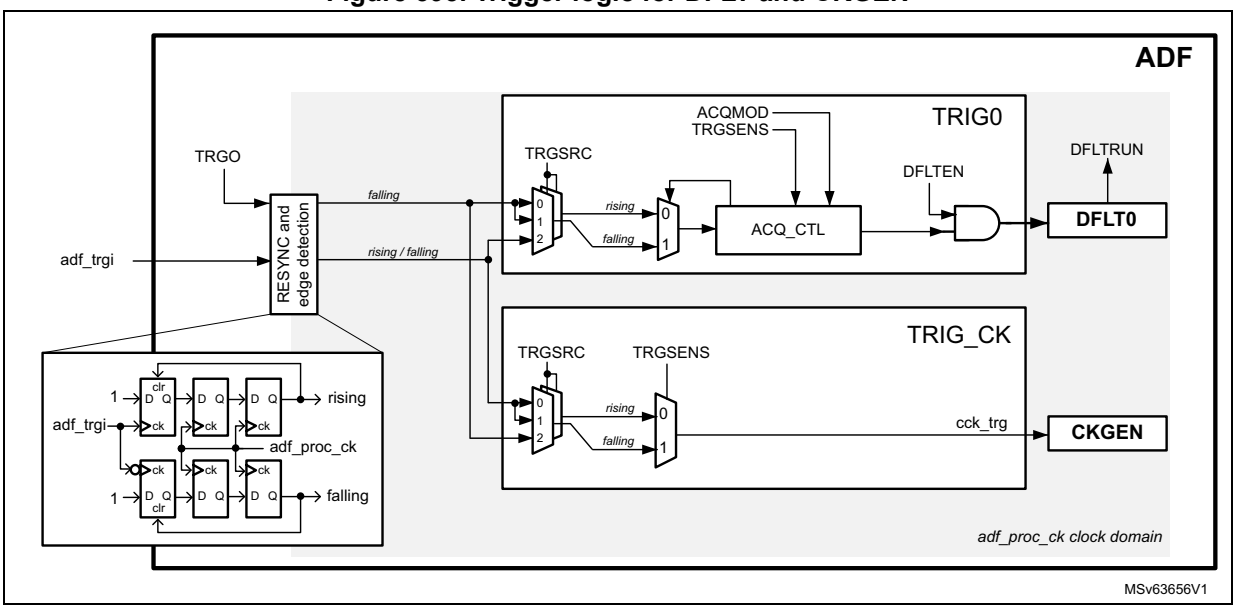
- asynchronous continuous acquisition mode
- asynchronous single-shot acquisition mode
- synchronous continuous acquisition mode
- synchronous single-shot acquisition mode
- window continuous acquisition mode

Note: To perform a data capture, the filter, the interface providing the data (SITF0 or ADCITF) and the CKGEN must be enabled. If needed, the ADF_CCK0 or ADF_CCK1 must be enabled as well.

The filter can be stopped immediately when DFLTEN is set to 0. This action resets the filter and flushes the RXFIFO. The DFLTACTIVE flag also goes back to 0 when the RXFIFO and the filter is reset.

Table 395 shows a simplified view of the trigger logic available for each filter and for the clock generator.

Figure 395. Trigger logic for DFLT and CKGEN



A block common to all TRIG blocks performs the rising and falling edges detection and the resynchronization of the input trigger to the adf_ker_ck clock domain. This implementation allows the application to use triggers with pulse width smaller than the adf_ker_ck period.

In synchronous modes, the TRIG block offers the possibility to select adf_trgi or TRGO bit as trigger sources. The TRGO bit is in the *ADF global control register (ADF_GCR)*.

The edge sensitivity can also be selected.

Asynchronous continuous acquisition mode

This mode allows the application to start a continuous acquisition by simply writing the DFLTEN bit to 1.

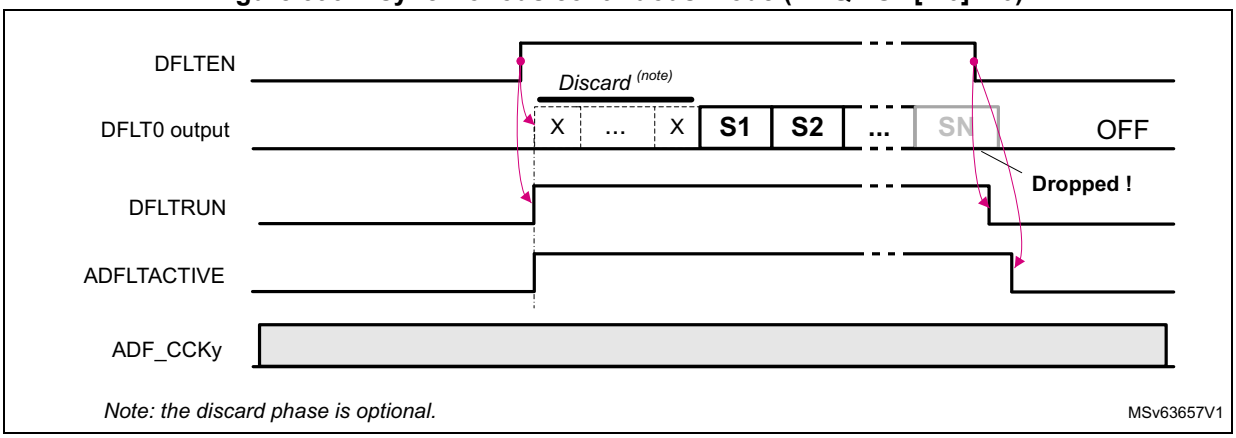
The asynchronous continuous acquisition mode is selected when ACQMOD[2:0] = 0.

The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN) so that the adf_proc_ck frequency is compatible with the targeted application (see examples in *Table 306*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration and set the ACQMOD[2:0] to 0.
4. Set to 1 the SITFEN bit of the serial data interface.
5. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
6. When DFLTEN is set to 1, the acquisition sequence starts immediately.

Figure 396 shows a simplified example of the samples generated by the DFLT0.

Figure 396. Asynchronous continuous mode (ACQMOD[2:0] = 0)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Asynchronous single-shot acquisition mode

This mode allows the application to start the acquisition of one sample by simply writing the DFLTEN bit to 1.

The asynchronous single-shot acquisition mode is selected when ACQMOD[2:0] = 001.

The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the adf_proc_ck frequency is compatible with the targeted application (see examples in *Table 306*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration, and set the ACQMOD[2:0] to 001.
4. Set to 1 the SITFEN bit.
5. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
6. When DFLTEN is set to 1, the filter provides one data to the RXFIFO and stops the acquisition.

To trigger a new acquisition, the application must:

1. Check that the previous acquisition is completed, by waiting that DFLTRUN = 0.
2. Set again DFLTEN to 1.

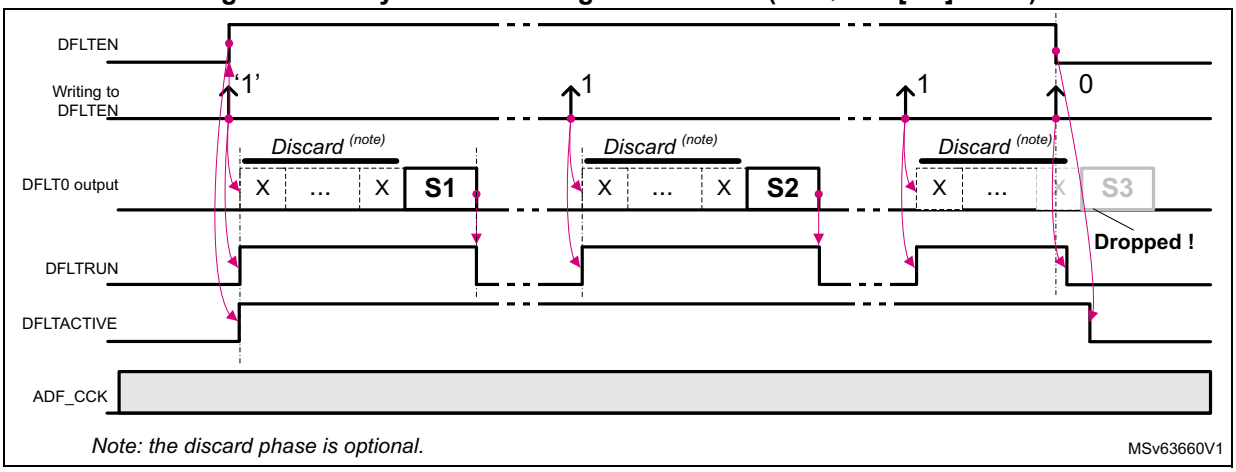
This sequence can be repeated every time a new data must be converted.

As shown in *Figure 396*, every time DFLTEN is set to 1, an acquisition sequence is triggered. The first samples provided by the filter can be discarded if needed. At the end of each conversion, the decimation counters and filter taps are reset, and the filter is ready to start a new conversion.

If DFLTEN is set to 0 while an acquisition is ongoing, the ongoing conversion is stopped (in the example, S3 is lost). This situation can be avoided with the following steps:

1. Wait for DFLTRUN = 0.
2. Read the sample from the RXFIFO.
3. Set DFLTEN to 0.

Figure 397. Asynchronous single-shot mode (ACQMOD[2:0] = 001)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Synchronous continuous acquisition mode

This mode allows the application to start a continuous acquisition by using one of the following trigger sources:

- adf_trgi signal
- TRGO bit

The Synchronous continuous acquisition mode is selected when ACQMOD[2:0] = 010.

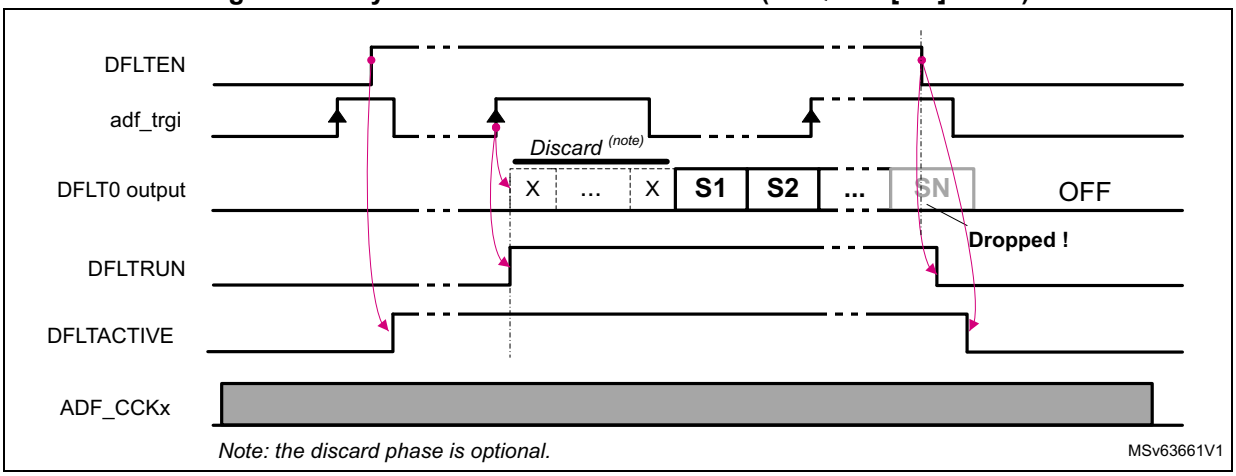
The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of adf_proc_ck clock is compatible with the targeted application (see examples in *Table 306*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration and set the ACQMOD[2:0] to 010.
4. Set to 1 the bit SITFEN.
5. Select the proper trigger source and sensitivity.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1.
8. When the trigger condition is met, the filter starts the acquisition.

The TRGSENS bit allows the selection of the trigger edge (rising or falling). The trigger is ignored if an acquisition is ongoing or if DFLTEN is set to 0.

Figure 398 shows a simplified example where the trigger logic is sensitive to a rising edge trigger (TRGSENS = 0). The first rising edge of the trigger signal is ignored because DFLTEN = 0. Then the next rising edge is taken into account and starts the acquisition. All other rising edges are ignored. The trigger logic is re-initialized when DFLTRUN goes back to 0.

Figure 398. Synchronous continuous mode (ACQMOD[2:0] = 010)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Synchronous single-shot acquisition mode

This mode allows the application to start a single acquisition by using one of the following trigger sources:

- adf_trgi signal
- TRGO bit

The Synchronous single-shot acquisition mode is selected when ACQMOD[2:0] = 011.

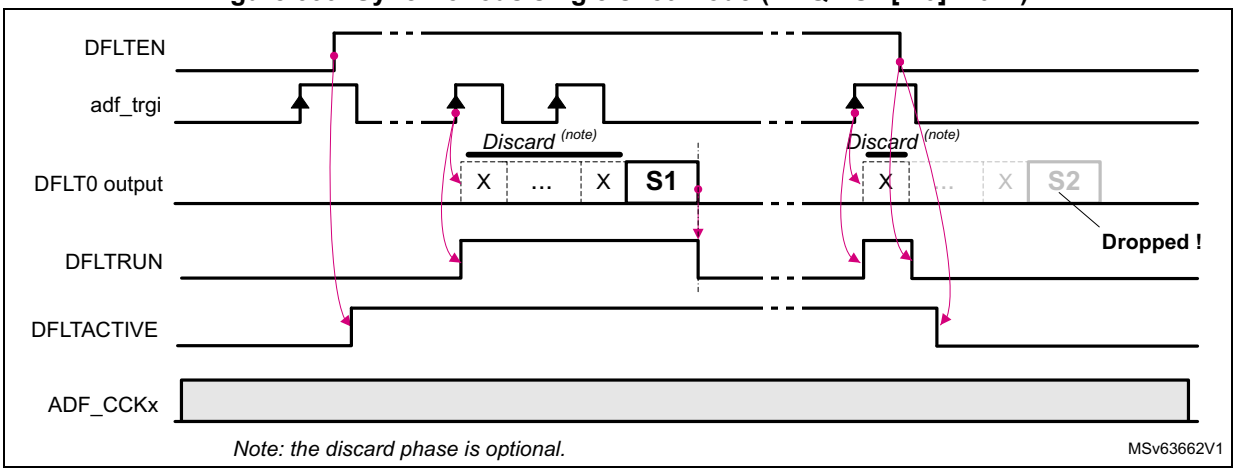
The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of adf_proc_ck clock is compatible with the targeted application (see examples in *Table 306*).
2. Enable the CKGEN and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration, and set the ACQMOD[2:0] to 011.
4. Set to 1 the SITFEN bit.
5. Select the proper trigger source and sensitivity.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1.
8. When the trigger condition is met, the filter starts the acquisition and provides one data to the RXFIFO, then the filter is ready to accept a new trigger.

TRGSENS allows the selection of the trigger edge (rising or falling). The trigger is ignored if an acquisition is ongoing, or if DFLTEN is set to 0.

Figure 399 shows a simplified example where the trigger logic is sensitive to a rising edge trigger (TRGSENS = 0). Every-time a trigger rising edge is detected with DFLTEN = 1, an acquisition sequence is triggered. The first samples provided by the filter can be discarded if needed. At the end of each conversion, the decimation counters and filter taps are reset. DFLTRUN is set to 0 and the filter is ready to start a new conversion.

Figure 399. Synchronous single-shot mode (ACQMOD[2:0] = 011)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Figure 398 shows a case where the DFLTEN is set to 0 while an acquisition is ongoing (the sample S2 is lost). This situation can be avoided with the following steps:

1. Wait for DFLTRUN = 0.
2. Read the sample from the RXFIFO.
3. Clear DFLTEN to 0.

Window continuous acquisition mode

This mode allows the application to start or stop a continuous acquisition controlled by consecutive edges of one of the following trigger sources:

- adf_trgi signal
- TRGO bit

The window continuous acquisition mode is selected when ACQMOD[2:0] = 100.

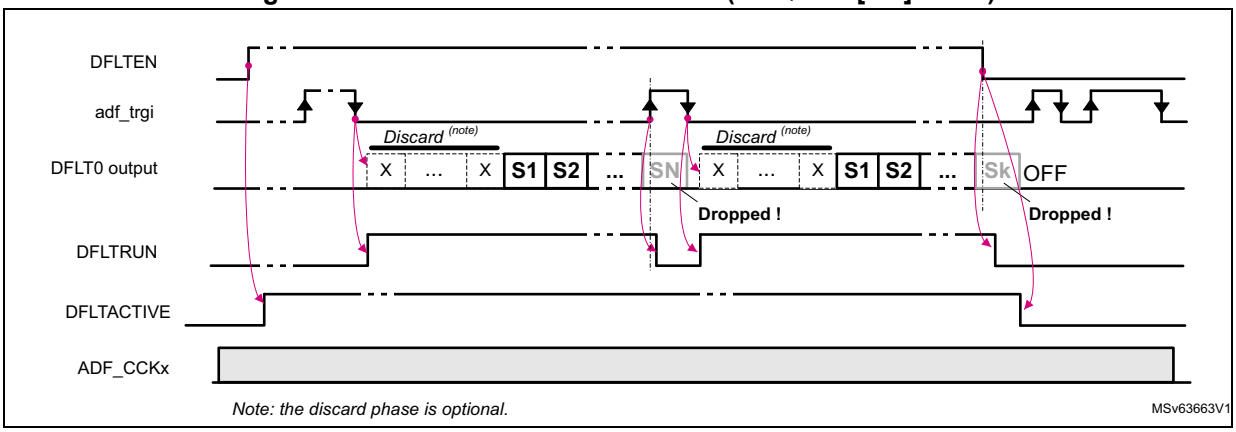
The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of adf_proc_ck clock is compatible with the targeted application (see examples in *Table 306*).
2. Enable the CKGEN and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter settings and set the ACQMOD[2:0] to 100.
4. Set to 1 the SITFEN bit.
5. Select the proper trigger source and sensitivity.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1.
8. If TRGSENS = 0, the acquisition starts on trigger rising edge and stops on trigger falling edge. If TRGSENS = 1, the acquisition starts on trigger falling edge and stops on trigger rising edge.

Note: The acquisition may restart if the trigger condition becomes again active.

Figure 400 shows a simplified example of window continuous acquisition mode, with TRGSENS = 1. Once DFLTEN is set to 1, the ADF waits for a falling edge on the selected trigger input. When the trigger condition is met, DFLTRUN goes to 1 and the acquisition starts. The acquisition stops if the ADF detects a rising edge on the selected trigger input. If DFLTEN is still set to 1, the ADF waits again for a falling edge on the selected trigger input.

Figure 400. Window continuous mode (ACQMOD[2:0] = 100)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Starting several filters synchronously

If the ADF is used with MDF instances (if present in the product), it is possible to start simultaneously the acquisition of all the filters. This synchronization capability depends on the way the triggers are connected in the product. Generally, an ADF is able to trigger MDF instances, if its adf_trgo signal is connected as trigger input to those blocks (see *Section 30.4.2: ADF pins and internal signals* to check trigger capabilities).

In the following programming example, one ADF has its adf_trgo signal connected to some MDFs. To start the acquisition of several filters synchronously, the following sequence must be performed (assuming that DFLTEN bits of the filters are set to 0):

On MDFs receiving the adf_trgo trigger:

1. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
2. Set to 1 the SITFEN bit of the requested data interfaces.
3. For each filter, set the acquisition mode to synchronous (ACQMOD[2:0] = 01x).
4. For each filter, set TRGSRC[3:0] in order to select the adf_trgo trigger input.
5. For each filter, set TRGSENS to 0 (rising edge).
6. For each filter, set DFLTEN to 1.

On the ADF generating the adf_trgo trigger:

1. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK[1:0] clocks.
2. Set to 1 the SITFEN bit of the requested data interfaces.
3. Set the acquisition mode to synchronous (ACQMOD[2:0] = 01x).
4. Set TRGSRC[3:0] to 0 (TRGO selected).
5. Set TRGSENS to 0 (rising edge).
6. Set DFLTEN to 1.
7. Read TRGO bit until it is read to 0.
8. Set TRGO to 1. Then the acquisition sequence for all selected filters starts immediately.

To trigger a new acquisition (in case of single-shot) the application must do the following:

1. Check that the previous acquisition is completed, by waiting DFLTRUN = 0.
2. Read TRGO until it is read to 0.
3. Set again the bit TRGO to 1.

Discarded samples

The ADF offers the possibility to program the amount of samples to be discarded after each restart:

- to avoid capturing samples affected by the impulse response of the filter
- to delay the acquisition of filters by a specific amount of samples

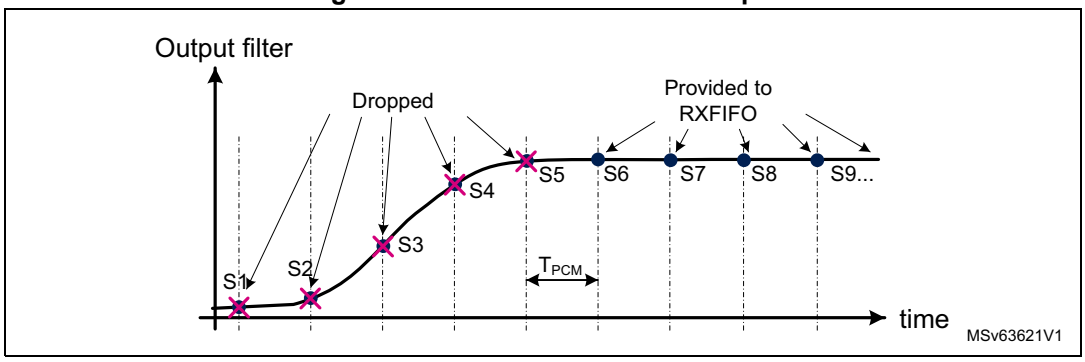
The discard function is controlled via NBDIS[7:0] as follows:

- When NBDIS[7:0] = 0, the discard function is disabled.
- When NBDIS[7:0] ≠ 0, the discard function is activated in one of the following condition:
  - when the DFLTEN bit goes to 1
  - every time an acquisition is started in (A)synchronous single-shot modes

Refer to *Figure 395* to *Figure 399*, and *Figure 401*.

In the example shown in *Figure 401*, the discard function is used to drop the first five samples provided by the digital filter (S1 to S5). The first sample transfered to the RXFIFO is S6.

Figure 401. Discard function example



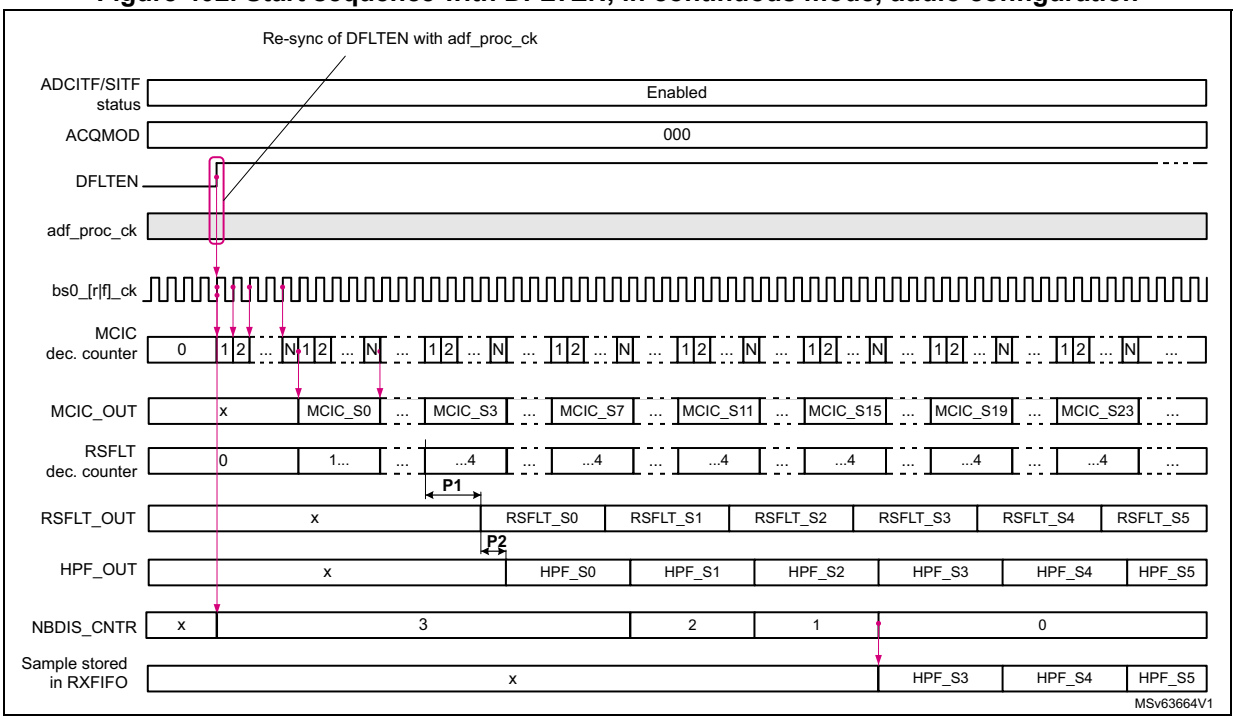
30.4.9 Start-up sequence examples

Figure 402 details a start of acquisition sequence of a digital filter triggered by DFLTEN (ACQMOD[2:0] = 0), with NBDIS[7:0] = 3 (three samples to discard before acquisition).

The DFLT0 is configured for audio application: MCIC, RSFLT and HPF activated.The data interface (SITF0 or ADCITF) is assumed to be already activated.

Note: NBDIS[7:0] is set on purpose to a small value to simplify the drawing.

Figure 402. Start sequence with DFLTEN, in continuous mode, audio configuration



The DFLTEN bit is re-sampled into the ADF processing clock domain. When DFLTEN is detected high, the filter chain is enabled, and the decimation counter of the MCIC filter is incremented at the rate of the bitstream clock.

When the MCIC decimation counter reached its programmed value N, a sample is available for the RSFLT.

The RSFLT processes all the samples provided by the MCIC, and delivers a sample to the HPF every-time it processes four samples (decimation by 4). The RSFLT needs up to 24 cycles of adf_proc_ck clock before delivering a sample (P1).

The HPF processes all the samples provided by the RSFLT, but the NBDIS function prevents the data writing in the RXFIFO as long as NBDIS_CNTR does not reach 0.

When NBDIS_CNTR reaches 0, the samples provided by the HPF are stored into the RXFIFO.

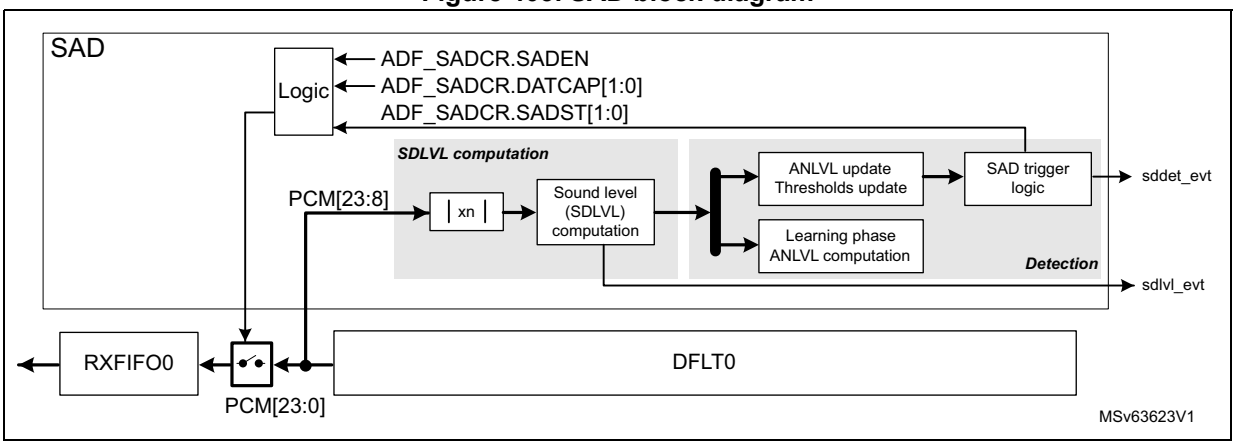
30.4.10 Sound activity detection (SAD)

The SAD is based on the computation of the ambient noise level (ANLVL) and of the short-term sound level (SDLVL). The SAD offers the following ways to detect a sound:

- when the SDLVL reaches a threshold referenced to the ambient noise level
- when the SDLVL reaches a fixed threshold
- when the ANLVL reaches a fixed threshold

As shown in *Figure 403*, the SAD takes the 16 MSB samples from the DFLT0 output.

Figure 403. SAD block diagram



The SAD is highly configurable, and the application can adjust several parameters:

- SAD detection behavior (SADMOD)
- number of samples used to compute the sound level (FRSIZE)
- number of frames used to compute the ambient noise level during the learning phase (LFRNB)
- slope of the ambient noise estimator (ANSLP)
- minimum expected ambient noise level (ANMIN)
- threshold level (SNTHR)
- threshold hysteresis (HYSTEN)
- hangover window in order to filter spurious transitions between DETECT and MONITOR states (HGOVR)
- data capture mode (DATCAP)

SAD detection behavior

The SAD can use the following ways to detect a sound, selected by SADMOD[1:0]:

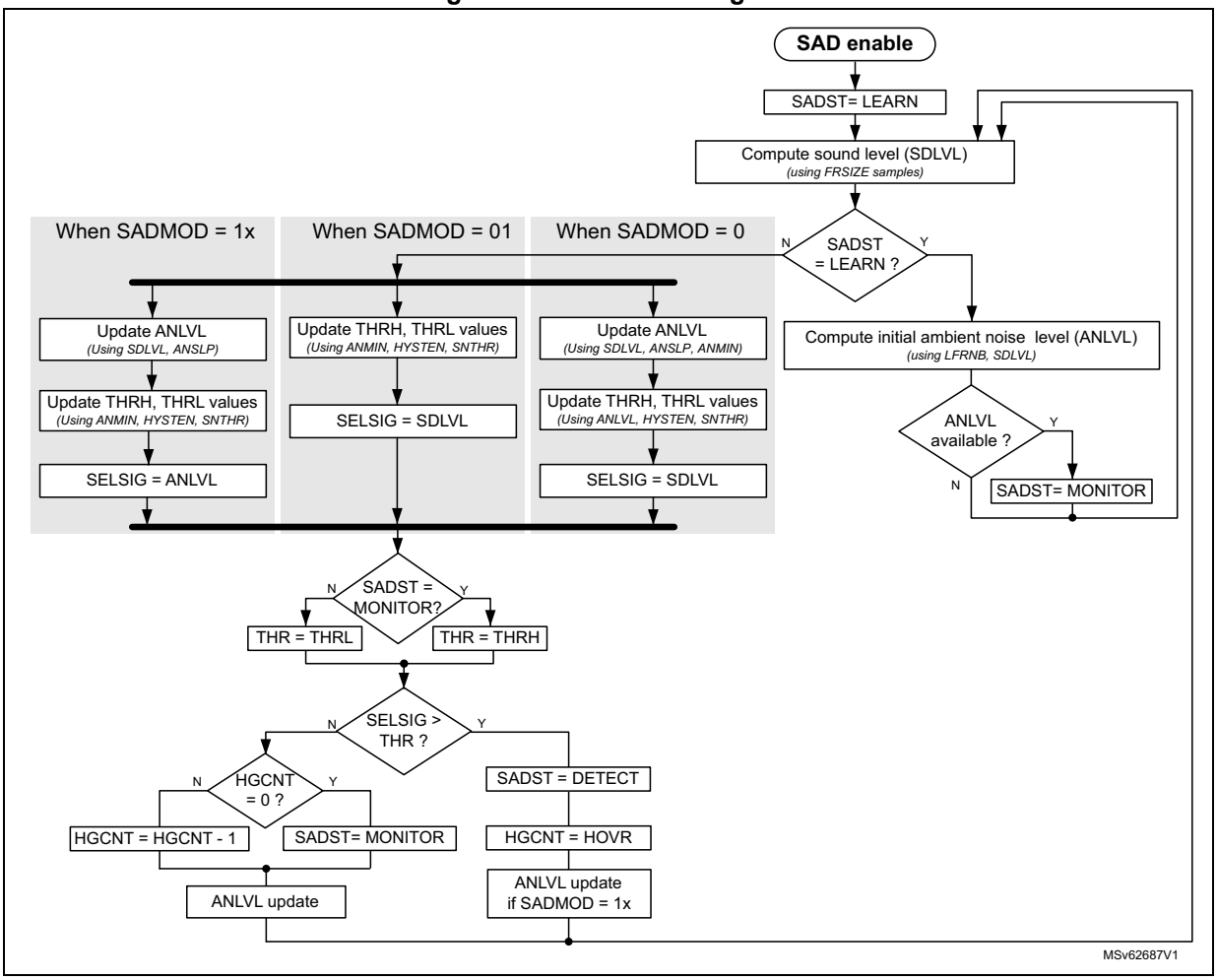
- When SADMOD[1:0] = 0, the SAD works like a voice-activity detection. In this mode, the SAD estimates the ambient noise level according to the computed sound level values. The threshold of the trigger is elaborated from the estimated ambient noise. Finally the current sound level is compared to this threshold. In a first approximation, the SAD triggers if the peak-to-average value of the input signal reaches a level defined by SNTHR[3:0].
- When SADMOD[1:0] = 01, the SAD compares the current sound level (SDLVL) to a fixed trigger value defined by the application via SNTHR[3:0] and ANMIN[12:0]. This mode allows a fast SAD reaction as the amount of samples used to compute the sound level can be configured via FRSIZE[2:0].
- When SADMOD[1:0] = 1x, the SAD compares the estimated ambient noise level (ANLVL) to a fixed trigger value defined by the application via SNTHR[3:0] and ANMIN[12:0]. This mode avoids unwanted triggers, due to peak levels, but the SAD reacts more slowly to an input signal variation. It is nevertheless possible to adjust the reaction time via FRSIZE[2:0] and ANSLP[2:0].

SAD states

As shown in *Figure 403*, the SAD works as follows:

1. When enabled (SADEN = 1), the SAD is first in LEARN state to perform a first estimation of the ambient noise level.
2. The SAD continuously computes the short-term sound level (SDLVL) using the samples provided by the DFLT0. The amount of samples used to compute the sound level is given by FRSIZE[2:0]. The samples processed by the DFLT0 can be transferred into the memory or not depending on DATCAP[1:0] value.
3. The initial ambient noise level (ANLVL) is computed using the consecutive sound level values. The application can define how much sound level values are used to perform the computation of this initial ambient noise estimation (LFRNB).
4. When the initial ambient noise level (ANLVL) is computed, the SAD switches to the MONITOR state.
5. Every time a new short-term sound level value is available, the SAD updates the ambient noise level and the thresholds according to the selected detection mode.
6. If the SAD triggers, then the following happens:
   – The SAD switches to DETECT state.
   – The sddet_evt event is asserted.
   – The adf_sad_det signal is set to high.
7. The hangover function insures that the DETECT state is maintained even if the sound level goes below the threshold level for a time given by HGOVR.

Figure 404. SAD flow diagram



Sound level computation (SDLVL)

Once enabled, the SAD computes continuously the sound level value. The sound level represents the average of the absolute value of an amount of PCM samples given by FRSIZE[2:0].

$$SDLVL = \frac{1}{N_{FRSIZE}} \times \sum_{n=1}^{N_{FRSIZE}} |PCM(n)|$$

where N_{FRSIZE} is the amount of PCM samples given by FRSIZE[2:0].

Ambient noise estimation (ANLVL)

The ambient noise level (ANLVL) is computed when SADMOD[1:0] is 00 or 10.

The ambient noise level is computed differently according to the state of the SAD as detailed below:

- ANLVL computation during the LEARN state

  Every time the SAD is enabled, a learning phase is initiated in order to estimate a first value of the ambient noise level. During this phase, the SAD cannot trigger.

  During the LEARN phase, the ambient noise level is computed as follows:

$$ANLVL = \frac{1}{N_{LFRNB}} \times \sum_{n=1}^{N_{LFRNB}} |SDLVL(n)|$$

  where N_{LFRNB} is the amount of frames given by LFRNB[2:0] bitfield.

- ANLVL computation during the MONITOR or DETECT state

  When the learning phase is completed, the SAD updates the ambient noise level in the following way:

  – The SAD computes the new possible values for the ambient noise level:

    ANLVL_UP = ANLVL * (1+2^(ANSLP-12))

    ANLVL_DN = ANLVL * (1-2^(ANSLP-10))

  – The ANLVL takes the ANLVL_DN value if the current sound level is lower than ANLVL_DN, otherwise ANLVL takes the value of ANLVL_UP.

    The ANLVL is not updated if the current sound level is higher than the threshold level, except if SADMOD[1:0] = 10.

  – When SADMOD[1:0] = 0, if the new ANLVL value is lower than ANMIN[12:0], ANLVL is replaced by ANMIN.

  The slope of the noise estimator can be adjusted to optimize the detection of the wanted signal. This slope is adjusted via ANSLP[2:0] in the *ADF SAD configuration register (ADF_SADCFGR)*.

Table 303 shows the allowed values according to the frame size and the sampling rate of the data observed by the SAD. The recommended values when the SADMOD[1:0] = 0 are the ones into the gray shaded cells.

Table 303. ANSLP values versus FRSIZE and sampling rates

FRSIZE	ANSLP values for Fs = 8 kHz			ANSLP values for Fs = 16 kHz		
	Slow(1)	typical(2)	Fast(3)	Slow(1)	typical(2)	Fast(3)
0 (8 samples)	0	1	2	-	0	1
1 (16 samples)	1	2	3	0	1	2
2 (32 samples)	2	3	4	1	2	3
3 (64 samples)	3	4	5	2	3	4
4 (128 samples)	4	5	6	3	4	5
5 (256 samples)	5	6	7	4	5	6
6 (512 samples)	6	7	-	5	6	7

1. The slow slope is equal to - 8.5 dB/s for the negative slope and + 2.1 dB/s for the positive slope.
2. The typical slope is equal to - 17.1 dB/s for the negative slope and + 4.2 dB/s for the positive slope.
3. The fast slope is equal to - 34.2 dB/s for the negative slope and + 8.5 dB/s for the positive slope.

The slopes can also be computed using the following formulas:

$$SLC_{UP} = 20 \times \frac{F_s}{F_{SIZE}} \times \log 10(1 + 2^{(ANSLP - 12)})$$

$$SLC_{DN} = 20 \times \frac{F_s}{F_{SIZE}} \times \log 10(1 - 2^{(ANSLP - 10)})$$

where F_S is the sampling rate of the stream observed by the SAD and F_{SIZE} is the frame size defined by FRSIZE[2:0].

Threshold computation

The way the threshold value is computed depends on SADMOD[1:0]:

- If SADMOD[1:0] = 0, THRH is obtained by multiplying the current ANLVL value with the gain defined in SNTHR[3:0].

  $$THR_H = ANLVL \times 10^{\frac{GdB_{SNTHR}}{20}}$$

  This threshold value is then compared to the current sound level (SDLVL).
- If SADMOD[1:0] = 01, THRH is obtained by multiplying the current ANMIN[12:0] with the gain defined by SNTHR[3:0].

  $$THR_H = ANMIN \times 10^{\frac{GdB_{SNTHR}}{20}}$$

  This threshold value is then compared to the current sound level (SDLVL).

- If SADMOD[1:0] = 1x, THRH is obtained by multiplying the current ANMIN[12:0] by 4.

$$THR_H = ANMIN \times 4$$

This threshold value is then compared to:

$$ANLVL \times 10^{\frac{GdB_{SNTHR}}{20}}$$

The hysteresis mode can be enabled to reduce the spurious transitions between MONITOR and DETECT states. In hysteresis mode (HYSTEN = 1), the following threshold values are used:

- THRH when the SAD is in MONITOR state.
- THRL when the SAD is in DETECT state.

Table 304 shows the thresholds values according to SNTHR.

Table 304. Threshold values according SNTHR(1)

SNTHR[3:0]	THR_H		THR_L		Comments
0	LVL + 3.5 dB	LVL x 1.5	LVL + 3.5 dB	LVL x 1.5	No hysteresis
1	LVL + 6.0 dB	LVL x 2	LVL + 3.5 dB	LVL x 1.5	Hysteresis of 2.5 dB
2	LVL + 9.5 dB	LVL x 3	LVL + 6.0 dB	LVL x 2	Hysteresis of 3.5 dB
3	LVL + 12.0 dB	LVL x 4	LVL + 9.5 dB	LVL x 3	Hysteresis of 2.5 dB
4	LVL + 15.6 dB	LVL x 6	LVL + 12.0 dB	LVL x 4	Hysteresis of 3.5 dB
5	LVL + 18.1 dB	LVL x 8	LVL + 15.6 dB	LVL x 6	Hysteresis of 2.5 dB
6	LVL + 21.6 dB	LVL x 12	LVL + 18.1 dB	LVL x 8	Hysteresis of 3.5 dB
7	LVL + 24.1 dB	LVL x 16	LVL + 21.6 dB	LVL x 12	Hysteresis of 2.5 dB
8	LVL + 27.6 dB	LVL x 24	LVL + 24.1dB	LVL x 16	Hysteresis of 3.5 dB
9	LVL + 30.1 dB	LVL x 32	LVL + 27.6dB	LVL x 24	Hysteresis of 2.5 dB

1. LVL must be replaced by ANLVL when SADMOD[1:0] = 0 and by ANMIN for other SADMOD[1:0] values.

When the hysteresis function is disabled, the SAD always use THRH.

Note: The hysteresis mode must not be used when SADMOD[1:0] = 1x.

Trigger logic

The signal compared to this threshold depends also on SADMOD[1:0].

The trigger condition is reached when the selected signal (SELSIG) is bigger than the threshold level.

If the trigger condition is met, the following happens:

- The SAD switches to DETECT state.
- The SAD refreshes the hangover counter with HGOVR.
- The sddet_evt event is asserted if the SAD transits from MONITOR to DETECT.
- The adf_sad_det signal is set to high.

The SAD remains in DETECT state as long as the trigger condition is met or the hangover down-counter is different from 0.

The sddet_evt event indicates when the SAD enters and/or exits the DETECT state. This event is used to generate an interrupt when a sound is detected or when a sound is no longer detected:

- When DETCFG = 0, the application receives an event only when the SAD enters the DETECT state.
- When DETCFG = 1, the application receives an event when the SAD enters or exits the DETECT state.

The adf_sad_det signal remains high as long as the SAD is in DETECT state.

The SAD also provides a flag indicating that a new sound level value is available (SDLVLF). The last computed sound level (SDLVL[14:0]) is available in the *ADF SAD sound level register (ADF_SADSDLVR)*, and the last computed ambient noise level (ANLVL[14:0]), in the *ADF SAD ambient noise level register (ADF_SADANLVR)*.

Note: The SAD can work even when the AHB clock is not present. In that case, the SAD does not update SDLVL[14:0] and ANLVL[14:0].

To get the latest valid SDLVL[14:0] and ANLVL[14:0] values, the application must read the ADF_SADSDLVR, and ADF_SADANLVR registers, when the SDLVLF flag goes high. This can be done in the following ways:

- by polling the SDLVLF flag:
  a) Clear the SDLVLF flag by writing SDLVLF to 1.
  b) Wait for SDLVLF = 1, by reading ADF_DFLTxISR.
  c) Read ADF_SADSDLVR and ADF_SADANLVR.
  d) Clear SDLVLF by writing it to 1.
  e) Go to step 2 if other values must to be read.
- by generating an interrupt:
  a) Read ADF_DFLTxISR.
  b) If SDLVLF = 1, read ADF_SADSDLVR and ADF_SADANLVR, and clear SDLVLF by writing it to 1.
  c) Handle other status flags and exit from ISR.

Sample transfer to memory

The SAD offers the following options to control the samples transfer from DFLT0 to the system memory:

- If DATCAP[1:0] = 1x, the samples are transferred into the system memory as soon as DFLT0 and SAD are enabled. The transfer does not depend on the SAD state.
- If DATCAP[1:0] = 01, the samples are transferred into the system memory when the SAD detects a sound (when the SAD is in DETECT state), assuming that DFLT0 and SAD are enabled.
- If DATCAP[1:0] = 0, the samples are not transferred into the memory. This mode can be used if the application only wants to observe but does not need samples for other processing.

Note: DATCAP[1:0] is taken into account only when the SADEN = 1. For example, if the SAD configuration is DATCAP[1:0] = 0, SADEN = DFLTEN = 1, and if the application sets now SADEN to 0, the samples provided by the DFLT0 are transferred to the RXFIFO.

Programming recommendations

To make the SAD function working properly, the ADF must be programmed as follows:

1. Provide the proper kernel clock (adf_ker_ck) to the ADF.
2. Configure the CKGEN and enable it.
3. Configure the SITF and enable it (note that microphones have a settling time of several milliseconds).
4. Configure the DFLT0. A typical setting is the following:
   – CIC5 with a decimation ratio of 12, 16 or 24
   – RSFLT with a decimation ratio of 4
   – HPF with HPFC = 2 or 3

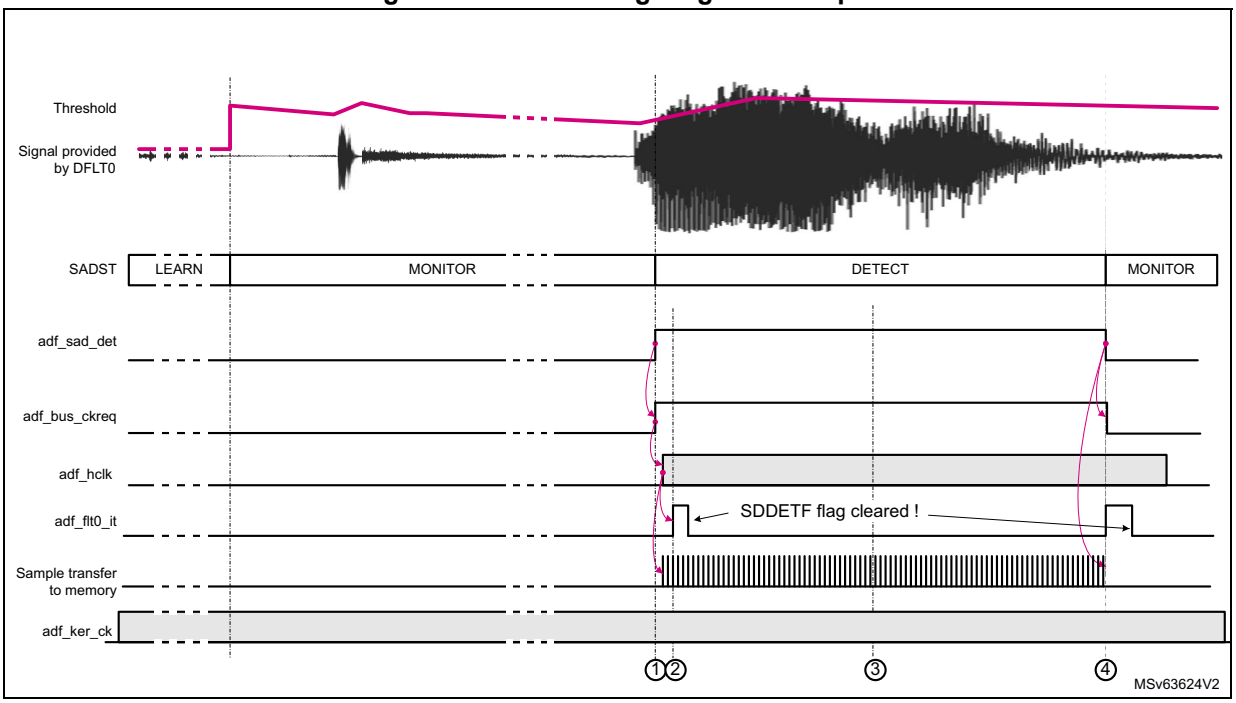
   For a very-low power implementation, the RSFLT can be bypassed.
5. Set SADEN to 0.
6. Wait for SADACTIVE = 0. If SADEN was previously enabled, this phase can take two periods of adf_hclk, and two periods of adf_proc_ck.
7. Configure the SAD as follows:
   – Set DATCAP[1:0] to 0, if the application does not want to store the samples into the system memory.
   – Set DATCAP[1:0] to 01, if the application wants to store the samples into the system memory only when the SAD detects a sound.
   – Set DATCAP[1:0] to 11, if the application wants to store the samples into the system memory continuously.
8. Configure the DMA (optional).
9. Enable the SAD.
10. Enable the DFLT0.

Figure 404 shows a simplified timing diagram when the SAD works with DATCAP[1:0] = 01.

Thanks to the kernel clock (adf_ker_ck), the SAD continuously monitors the audio signal provided by the DFLT0. The threshold is also continuously updated according to the ambient noise level estimation.

- When the SAD detects a sound higher than the programmed threshold (1), the ADF requests the bus clock (adf_bus_ckreq asserted).
- When the bus clock is available (see 2 in *Figure 404*) then:
  – The data transfer to the memory is triggered.
  – The event interrupt (adf_evt_it) can be generated.
- In this example, the event interrupt (adf_evt_it) is used to wake up the application. The interrupt line is released by clearing SDDETF by writing 1 to it.
- As long as the SAD remains in DETECT state, the application waits to get enough samples and calls, for example the keyword recognition algorithm (see 3 in *Figure 404*).
- In the case shown in *Figure 405*, the SAD state (SADST) goes back to MONITOR before the keyword is recognized. If DETCFG is set to 1, an event signals when the SAD goes back to MONITOR state. The SAD stops the transfer of samples into the memory and the application can clean up the receive buffer for the next detection (see 4 in *Figure 404*).

Figure 405. SAD timing diagram example



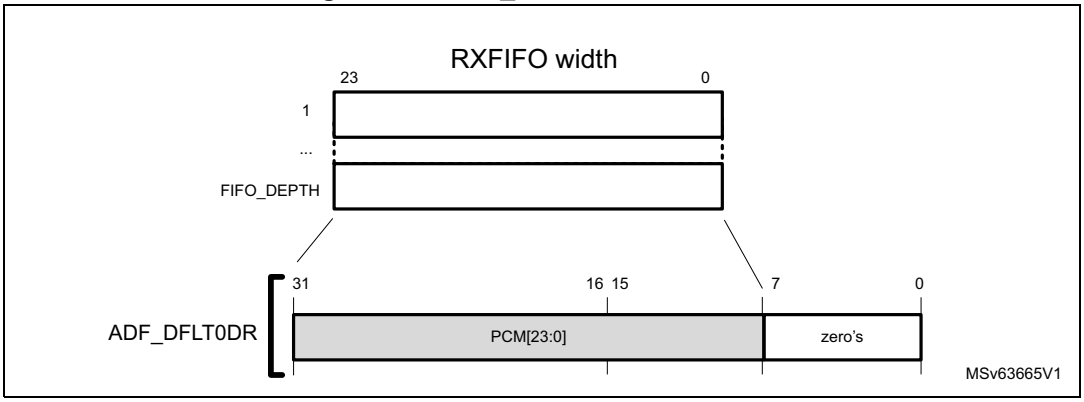
30.4.11 Data transfer to memory

Data format

The samples processed by DFLT0 are stored into a RXFIFO. The application can read the samples stored into these FIFOs via the *ADF digital filter data register 0 (ADF_DFLT0DR)*. The samples inside this register are signed and left aligned. The bit 31 always represents the sign.

The ADF provides 24-bit left-aligned data. Performing a 16-bit access to ADF_DFLT0DR allows the application to get the 16 most significant bits. Performing a 32-bit access to ADF_DFLT0DR allows the application to get a 24-bit data size.

Figure 406. ADF_DFLTxDR data format



Data resynchronization

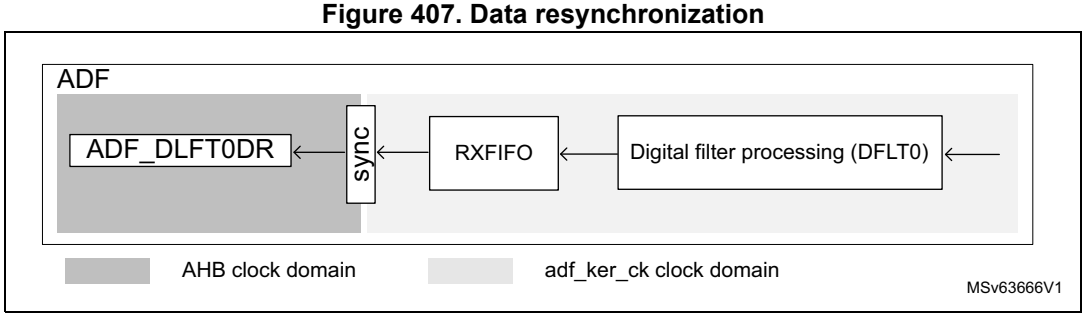
The samples stored into the RXFIFO can be transferred into the memory by using either DMA requests or interrupt signaling.

Note: The RXFIFO is located into the adf_ker_ck clock domain, while ADF_DFLT0DR *is located into the adf_hclk (AHB) clock domain.*

When the AHB clock is available, if ADF_DFLT0DR is empty and if a sample is available into the RXFIFO, this sample is transferred into ADF_DFLT0DR.

The sample transfer from the RXFIFO to ADF_DFLT0DR takes two periods of the AHB clock (adf_hclk) and two periods of the adf_ker_ck clock. The ADF inserts automatically wait-states if the application performs a read operation of ADF_DFLT0DR while the transfer of the new sample from the RXFIFO to ADF_DFLT0DR is not yet completed.

Figure 407. Data resynchronization



Data transfer

The content of the RXFIFO can be transfered to the memory either by using a DMA channel or interrupt services.

Both single and burst, DMA transfers are supported by the ADF, but the application has to care about the following points:

- The RXFIFO must contain at least the same amount of samples than the burst size.
- The burst mode efficiency may be reduced due to the data resynchronization explained in the previous section.

Note: The burst mode is not available in all products (see the DMA section to check if the product supports it).

In addition, the application can select the RXFIFO threshold (FTH bit) in order to trigger the data transfer: a data transfer can be triggered as soon as the RXFIFO is not empty, or when the RXFIFO is half-full (containing depth/2 samples).

For the DMA transfer, as soon as one of the RXFIFO reaches the threshold level, the DMA request is asserted in order to ask for data transfer. Successive DMA requests are performed as long as the RXFIFO is not empty.

The DMA mode of the RXFIFO is enabled via the DMAEN bit in ADF_DFLT0DR.

For the interrupt signaling, the following cases must be considered:

- If FTH = 0, as soon as a data is available in ADF_DFLT0DR, the FTHF is set, allowing the generation of an interrupt. FTHF is released as soon as ADF_DFLT0DR is read.
- If FTH = 1, as soon as the RXFIFO reaches the threshold level and a data is available in ADF_DFLT0DR, the FTHF is set, allowing the generation of an interrupt. FTHF is released as soon as one data is read. FTHF is set again if the threshold condition is

met again. In this mode, every time an interrupt occurs, the application is supposed to read FIFO_SIZE/2 data.

RXFIFO overrun

A RXFIFO overrun condition is detected when the RXFIFO is full, and a new sample from the DFLT0 must be written.

In this case, DOVRF is set and the new sample is dropped. When the RXFIFO has at least one location available, the new incoming sample is written into the RXFIFO.

Figure 408 shows an example based on a RXFIFO depth of four words and FTH set to 1, so that FTHF goes to 1 when the RXFIFO is half-full.

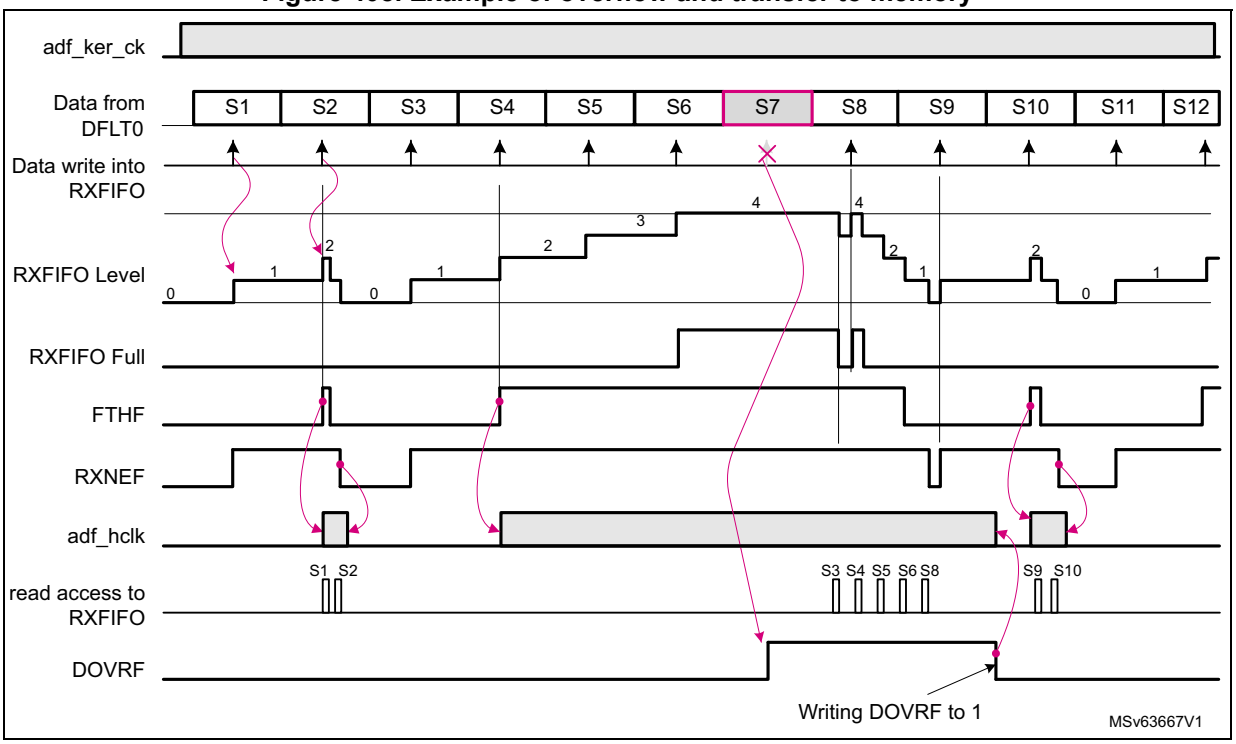
The S7 sample is lost due to an overrun: the RXFIFO is full while S7 must be written into the RXFIFO. The S7 write operation is not performed. DOVRF is set to 1 at the moment where the write operation was expected. The overflow event remains to 1 as long as it is not cleared by the application.

In this example, DOVRIE is set to 1 to have an interrupt if an overrun condition is detected.

After the S7 sample, the application manages to read data from the RXFIFO, and the ADF can write the S8 sample and consecutive. Later, the application clears DOVR, allowing the detection of a new overrun situation.

In the adf_hclk line, the gray boxes indicate that the ADF requested the AHB clock. *Figure 408* shows the AHB clock available only when the ADF requests it. In real applications, the AHB clock may also be present if the ADF does not request it.

Figure 408. Example of overflow and transfer to memory



30.4.12 Autonomous mode

The ADF can work even if the AHB bus clock is not available (Stop modes). The ADF uses the AHB clock only for the register interface. All the processing part is clocked with the kernel clock.

In Stop mode, the ADF receives a kernel clock if the following conditions are met:

- The ADF autonomous mode is enabled in the RCC.
- The selected kernel clock source is taken from an oscillator available in Stop mode.

In Stop mode, the ADF receives the AHB clock if the following conditions are met:

- The ADF autonomous mode is enabled in the RCC.
- The ADF requests the AHB clock in the following situations:
  - when the ADF must transfer data into memory via the DMA

    The data is directly transferred to the SRAM thanks to the DMA while the product remains in Stop mode. The AHB clock request is maintained until the DMA transfer is completed.
  - when the ADF needs to generate an interrupt

    An interrupt generally wakes up the device from Stop mode, as an action from the application is needed. Once the AHB clock is available, the interrupt is generated.The AHB clock request is maintained as long as an enabled interrupt flag is still active.

30.4.13 Register protection

The ADF embeds some hardware protection to prevent invalid situations.
Table 305 shows the list of write-protected and unprotected fields.

Table 305. Register protection summary

Registers	Unprotected fields	Write-protected fields	Write-protection condition
ADF global control register (ADF_GCR)	TRGO	-	DFLTACTIVE0 = 1
ADF clock generator control register (ADF_CKGCR)	CKGDEN CCK0EN CCK1EN	PROCDIV[6:0], CCKDIV[3:0], CKGMOD, TRGSRC[3:0], TRGSENS, CCK[1:0]DIR	CKGACTIVE = 1
ADF serial interface control register 0 (ADF_SITF0CR)	SITFEN	STH[4:0], SITFMOD[1:0], SCKSRC[1:0]	SITFACTIVEx = 1
ADF bitstream matrix control register 0 (ADF_BSMX0CR)	-	BSSEL[4:0]	DFLTACTIVEx = 1
ADF digital filter control register 0 (ADF_DFLT0CR)	DFLTEN	NBDIS[7:0], TRGSRC[3:0], TRGSENS, FTH, DMAEN, SNPSFMT, ACQMOD[2:0]	DFLTACTIVEx = 1
ADF digital filer configuration register 0 (ADF_DFLT0CICR)	SCALE[5:0]	MCICD[8:0], CICMOD[2:0], DATSRC[1:0]	
ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)	-	All fields	
ADF delay control register 0 (ADF_DLY0CR)	-	SKPDLY[6:0]	SKPBF = 1

Table 305. Register protection summary (continued)

Registers	Unprotected fields	Write-protected fields	Write-protection condition
ADF DFLT0 interrupt enable register (ADF_DFLT0IER)	All fields	-	-
ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)	All fields	-	-
ADF SAD control register (ADF_SADCR)	SADEN	FRSIZE[2:0], DETCFG, SADMOD[1:0], HYSTEN, DATCAP[1:0]	SADACTIVE = 1
ADF SAD configuration register (ADF_SADCFGR)	-	ANMIN[12:0], SNTHR[3:0], HGOVR[3:0], LFRNB[2:0], ANSLP[2:0]	SADACTIVE = 1

All the ADF processing is performed in the adf_proc_ck clock domain. For that reason, enabling or disabling an ADF sub-block may take some time due to the resynchronization between the AHB clock domain and the adf_proc_ck clock domain. XXXACTIVE flags are available to allow the application to check that the synchronization between the two clock domains is completed.

To change a write-protected bitfield, the application must follow this sequence:

1. Set the enable bit of the sub-block to 0.
2. Wait for corresponding flag XXXACTIVE = 0.
3. Modify the wanted fields.
4. Set the enable bit of the sub-block to 1.

Refer to the description of each sub-block for details.

30.5 ADF low-power modes

Table 306. Effect of low-power modes on ADF

Mode	Description
Sleep	No effect. ADF interrupts cause the device to exit the Sleep mode.
Stop(1)	The ADF registers content is kept. Thanks to the kernel clock request feature, if the ADF is clocked by an internal oscillator available in Stop mode, the ADF remains active. The interrupts cause the device to exit Stop mode.
Standby	The ADF is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 30.3: ADF implementation* for details about Stop modes supported by the ADF.

30.6 ADF interrupts

To increase the CPU performance, the ADF offers an interrupt line (adf_flt0_it), sensitive to several events.

Note: The status flags are available even if the corresponding interrupt enable flag is not enabled.

The interrupt interface is controlled via the *ADF DFLT0 interrupt enable register (ADF_DFLT0IER)* and the *ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)*.

Figure 409. ADF interrupt interface

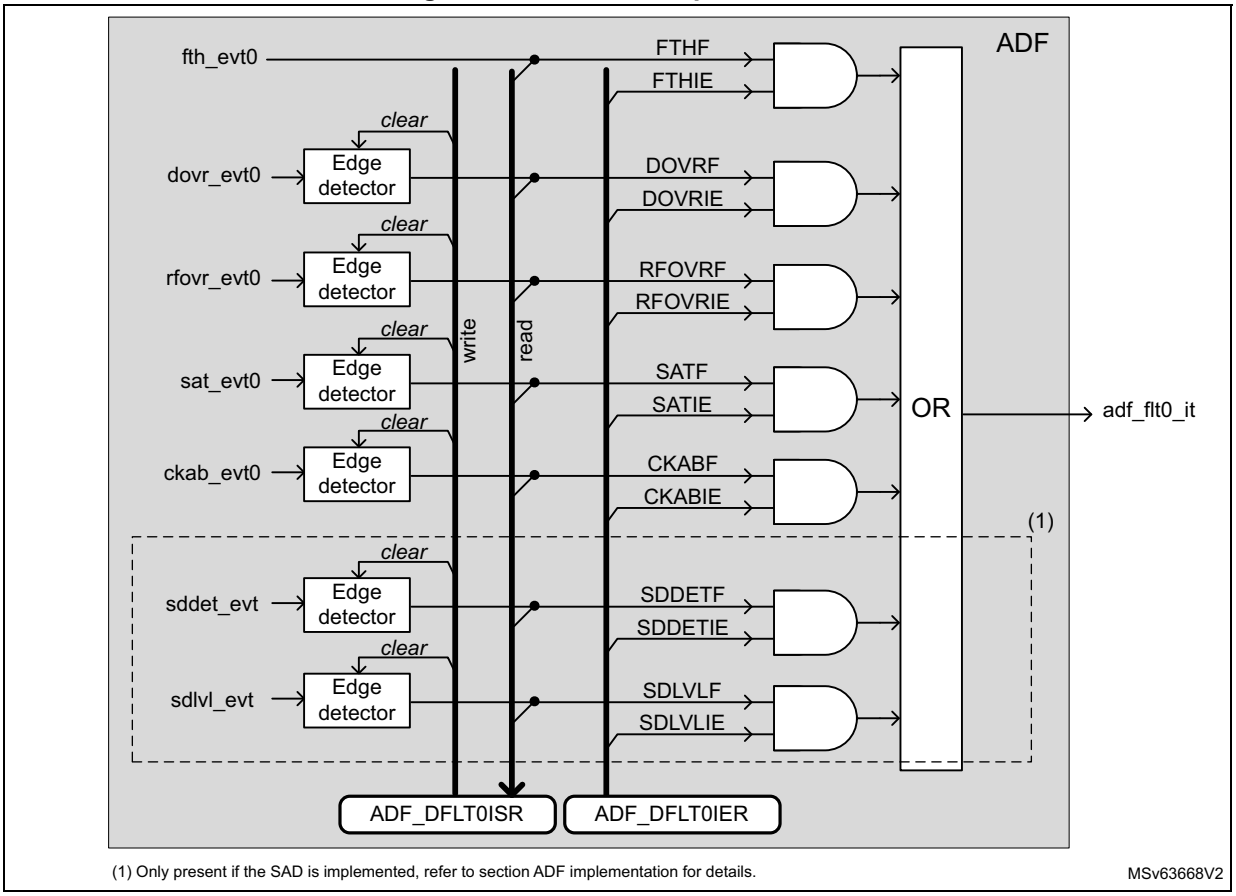


Table 307 shows which interrupt line is affected by which event, and how to clear and activate each interrupt/event.

Table 307. ADF interrupt requests

Interrupt vector	Interrupt event	Event flag	Event/interrupt clearing method	Exit Sleep mode	Exit Stop modes(1)	Exit Standby mode
ADF_FLT0(2)	RXFIFO threshold reached	FTHF	Read ADF_DFLT0DR until RXFIFO level is lower than the threshold.	Yes	Yes	No
	RXFIFO overrun	DOVRF	Write DOVRF to 1.			
	RSFLT overrun	RFOVRF	Write RFOVRF to 1.			
	Saturation detection	SATF	Write SATF to 1.			
	Channel clock absence detection	CKABF	Write CKABF to 1.			
	SAD: sound detected	SDDETF	Write SDDETF to 1.			
	SAD: sound level value available	SDLVLF	Write SDLVLF to 1.			

1. Refer to *Section 30.3: ADF implementation* for details.
2. ADF_FLT0 vector corresponds to the assertion of adf_flt0_it signal.

30.7 ADF application information

30.7.1 ADF configuration examples for audio capture

Table 308 gives some examples of the ADF settings for the digital microphones, focusing on 16 and 48 kHz output data rate.

Configurations #1 and #2 are for very low-power use-cases and have a reduced signal-to-noise ratio. The user must also insure that the selected digital microphone can work properly at 512 kHz. These configurations can be used for sound detection. The RSFLT is not used to reduce as much as possible the frequency of the kernel clock (adf_ker_ck).

Configurations #3, #4, #9, #10, #11 give signal-to-noise ratios around 115 dB, with an ideal microphone model, with a sinus signal of 997 Hz. Using the RSFLT allows a good control on the in-band ripple, and a good image rejection.

Configurations #7, #8, #10 give signal-to-noise ratio around 120 dB, with an ideal microphone model, using a sinus signal of 997 Hz.

Table 308. Examples of ADF settings for microphone capture

Configuration	adf_ker_ck (MHz)	PROCDIV + 1	CCKDIV + 1	CIC order (1)	MCICD + 1	SCALE	RSFLTBYP	RSFLTD	HPFBYP	-	adf_proc_ck (MHz)	Total dec. ratio	F_{RS} (kHz)	F_{ADF_CCKx} (MHz)	F_{PCM} (kHz)
#1	1.024	1	2	4	64	0x2D (- 8.5 dB)	1	x	x	=>	1.024	64	-	0.512	8
#2		1	2	5	32	0x2B (- 14.5 dB)						32	-		16
#3		1	2	5	16	0x01 (+ 3.5 dB)	0	0				64	32		8
#4	2.048	1	2	5	16	0x01 (+ 3.5 dB)	0	0			2.048	64	64	1.024	16
#5	3.072	1	6	5	8	0x0B (+ 33.6 dB)	0	0			3.072	32	32	0.512	8
#6		2	2	5	12	0x06 (+ 18.1 dB)						48	64	0.768	16
#7		1	2	5	24	0x2C (- 12 dB)						96		1.536	
#8	4.096	1	2	5	32	0x27 (- 26.6 dB)	0	0			4.096	128	64	2.048	16
#9	6.144	3	2	5	16	0x02 (+ 6.0 dB)	0	0			2.048	64	64	1.024	16
#10		2	2	5	24	0x2C (- 12 dB)					3.072	96		1.536	
#11		1	2	5	16	0x01 (+ 3.5 dB)					6.144	64	192	3.072	48
#12	7.680	1	2	5	20	0x2E (- 6.0 dB)	0	0			7.680	80	192	3.840	48

1. CICMOD = 100 for CIC order equal to 4. CICMOD = 101 for CIC order equal to 5.

30.7.2 Programming examples

Example 1

This example describes the programming of ADF for the capture of a signal coming from a digital microphone, using only the CIC4, with a decimation of 48, assuming that the kernel clock is 1.536 MHz. Typically, this configuration can be used to detect sound using the SAD.

Table 309. Programming sequence (CIC4)

Operations	Comments
Adjust the proper kernel clock frequency via the RCC	Assuming that the RCC is programmed to provide a kernel clock (adf_ker_ck) of 1.536 MHz coming from a RC oscillator.
Select the proper ADF kernel clock source via the RCC	Refer to the RCC of the product.
Enable the ADF clocks via the RCC	Refer to the RCC of the product.
Reset the ADF via the RCC	Refer to the RCC of the product.
AFMUX programming	Program the AFMUX to select ADF_SD0 and ADF_CCK0 functions.
Enable ADF processing clock: ADF_CKGCR = 0x0001 0023	PROCDIV = 0 (bypass): adf_proc_ck frequency is 1.536 MHz. CCKDIV = 1 (division by 2): ADF_CCK0 clock frequency is 768 kHz. The ADF_CCK0 pin is set in output and generates a clock so that the microphone can exit from low-power mode.
Serial interfaces configuration: ADF_SITF0CR = 0x0000 1F01	SCKSRC = 0 to select ADF_CCK0 as serial clock. SIFTMOD = 0 to select LF_MASTER SPI mode. Clock absence feature is not working in this mode. The serial interface is enabled.
Bitstream matrix configuration: ADF_BSMX0CR = 0x0000 0000	DFLT0 filter takes the bitstream of SITF0, sampled on rising edge
Filters configuration (CIC): ADF_DFLT0CICR = 0x0040 2F40	SCALE = 0x04 (12 dB): RSFLT is not enabled. Note that the gain is 8.5 dB, higher than recommended in order to improve signal accuracy for the SAD. Saturation is not an issue in this case, as only the detection of a signal much lower than the full scale is needed. MCICD = 0x2F (decimation by 48) CICMOD = 4 to select a $Sinc^4$ DATSCR = 0 to select data coming from BSMX
Filters configuration (RSFLT and HPF): ADF_DFLT0RSFR = 0x0000 0301	HPFC = 3: cut-off frequency of 16kHz * 0.0095 = 152 Hz HPFBYP = 0: HPF not bypassed RSFLTBYP = 1: RSFLT bypassed
Micro delay adjust: ADF_DLY0CR = 0x0000 0000	Not used in this example
Enable interrupt events: ADF_DFLT0IER = 0x0000 1000	Enable the interrupt events the application wants to handle. In this example, SDDETIE is set to 1 to have an interrupt if a sound is detected.

Table 309. Programming sequence (CIC4) (continued)

Operations	Comments
SAD setting: ADF_SADCR = 0x0000 1400	SADMOD = 1: Trigger based on sound level values FRSIZE = 4: Sound level computed on 128 samples HYSTEN = 0: No hysteresis DETCFG = 0: interrupt generated when the SAD enters in DETECT state DATCAP = 0: samples not transferred to memory SADEN = 0: SAD not yet enabled
SAD setting: ADF_SADCFGR = 0x0025 0001	ANMIN = 0x25: Sound threshold of 37 * 10^SNTHR/20 = 74 LSB(1) HGOVR = 0: four frames (4 x 128 = 512 samples) LFRNB = 0: Learning phase set to two frames. Not useful in this mode. ANSLP = 0: Not used in this mode SNTHR = 1: Threshold set to 6 dB (see ANMIN)
Clear status flags: ADF_DFLT0ISR = 0x0000 0FFF	Clear all the status flags before running the ADF
Enable the SAD: ADF_SADCR = 0x0000 1401 Wait for SADACTIVE flag = 1	Enable first the SAD, in order to be sure that data transfer to memory are blocked by the SAD.
Enable the filter: ADF_DFLT0CR = 0x0000 0001	NBDIS = 0: no samples discarded ACQMOD = 0: Asynchronous continuous acquisition mode DMAEN = 0: DMA interface not used DFLTEN = 1: Filter enabled

1. SNTHR and ANMIN values are computed using the same approach than in *Threshold programming with SADMOD = 01* (detection of a sound higher than 63 dBSPL).

Example 2

This example describes the programming of ADF for the capture of a signal coming from a digital microphone, using the CIC5, and the RSFLT, with a total decimation of 64.

Table 310. Programming sequence (CIC5)

Operations	Comments
Adjust the proper kernel clock frequency via the RCC	Assuming that the RCC is programmed to provide a kernel clock (adf_ker_ck) of 6.144 MHz
Select the proper ADF kernel clock source via the RCC	Refer to the RCC of the product.
Enable the ADF clocks via the RCC	Refer to the RCC of the product.
Reset the ADF via the RCC	Refer to the RCC of the product.
AFMUX programming	Program the AFMUX to select ADF_SD0 and ADF_CCK0 functions.
Enable ADF processing clock: ADF_CKGCR = 0x0201 0023	PROCDIV = 2 (division by 3): adf_proc_ck frequency is 6.144 MHz. CCKDIV = 1 (division by 2): ADF_CCK0 clock frequency is 1.024 MHz. The ADF_CCK0 pin is set in output and generates a clock so that the microphone can exit from low-power mode.

Table 310. Programming sequence (CIC5) (continued)

Operations	Comments
Serial interfaces configuration: ADF_SITF0CR = 0x0000 1F01	SCKSRC = 0 to select ADF_CCK0 as serial clock. SIFTMOD = 0 to select LF_MASTER SIPI mode. Clock absence feature is not working in this mode. The serial interface is enabled.
Bitstream matrix configuration: ADF_BSMX0CR = 0x0000 0000	DFLT0 filter takes the bitstream of SITF0, sampled on rising edge
Digital filter control: ADF_DFLT0CR = 0x0000 0006	NBDIS = 0: no samples discarded ACQMOD = 0: Asynchronous continuous acquisition mode DMAEN = 1: DMA interface not used DFLTEN = 0: Filter disabled FTH = 1: RXFIFO half-full
Filters configuration (CIC): ADF_DFLT0CICR = 0x0010 0F50	SCALE = 0x01(+3.5 dB): Output limited to 22 bits due to RSFLT. MCICD = 0xF (decimation by 16) CICMOD = 5 to select a $Sinc^5$ DATSCR = 0 to select data coming from BSMX
Filters configuration (RSFLT and HPF): ADF_DFLT0RSFR = 0x0000 0200	HPFC = 2: cut-off frequency of 16 kHz * 0.0025 = 40 Hz HPFBYP = 0: HPF not bypassed RSFLTBYP = 0: RSFLT not bypassed RSFLTD = 0: Decimation by 4
Micro delay adjust: ADF_DLY0CR = 0x0000 0000	Not used in this example
Enable interrupt events: ADF_DFLT0IER = x	Enable the interrupt events the application wants to handle.
Clear status flags: ADF_DFLT0ISR = 0x0000 0FFF	Clear all the status flags before running the ADF
Program the DMA Enable the DMA	The DMA must be programmed in order to read the data inside the ADF_DLT0DR register every time a DMA request is generated.
Start acquisition: ADF_DFLT0CR = 0x0000 0007	The ADF starts to filter data.

30.7.3 Connection examples

Figure 409 shows simple connection examples of the ADF to external sensors.

- Picture on the left: two digital microphones connected to the ADF

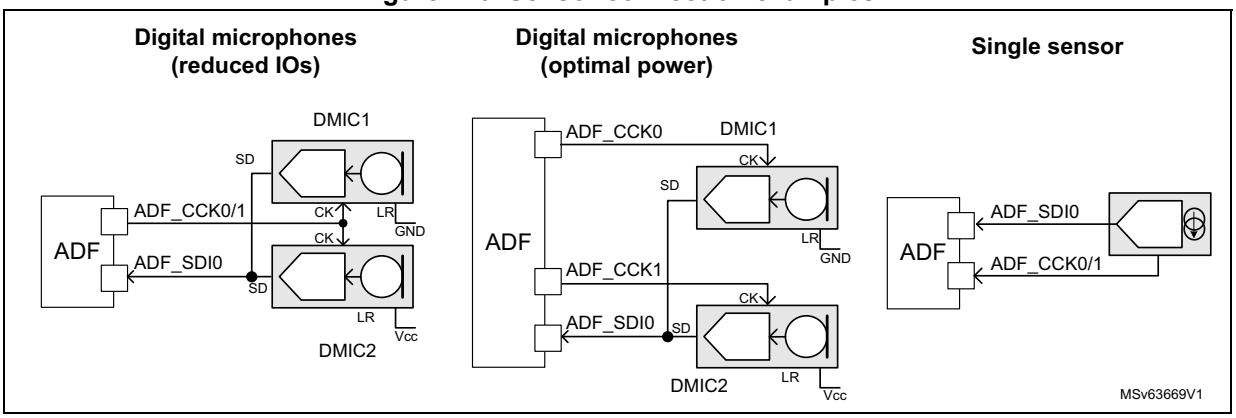
  In this connection, the amount of connections is optimized; DMIC1 and DMIC2 are sharing the same data line, and the same clock line. BSMX allows the application to connect the digital filter either to DMIC1 or to DMIC2. In this configuration when one of the microphone is used, the other is activated as well, as they share the same clock.
- Picture in the center: two digital microphones connected to the ADF

  In this connection, DMIC1 and DMIC2 are sharing the same data line, but have a dedicated clock line. BSMX allows the application to connect the filter either to DMIC1 or to DMIC2. When the application wants to use a microphone, it is possible to keep the other in low-power mode by forcing its clock line to 0 (CCKyEN = 0, CCKyDIR = 1).

- Picture on the right: single sensor connected to the ADF

  It is also possible to configure the CCK0 and CCK1 pins to input in order to connect sensors providing the clock.

Figure 410. Sensor connection examples



30.7.4 Global frequency response

Figure 410 shows the global frequency response for a 16 kHz audio signal with a digital microphone working at 1.024 MHz. The filter configuration is the following:

- CIC order 4 or 5, with a decimation ratio of 16
- RSFLT enabled, with a decimation ratio of 4
- HPF enabled with a cut-off frequency of 40 Hz

Figure 411 shows the theoretical frequency response using a CIC4 and a CIC5.

Figure 411. Global frequency response

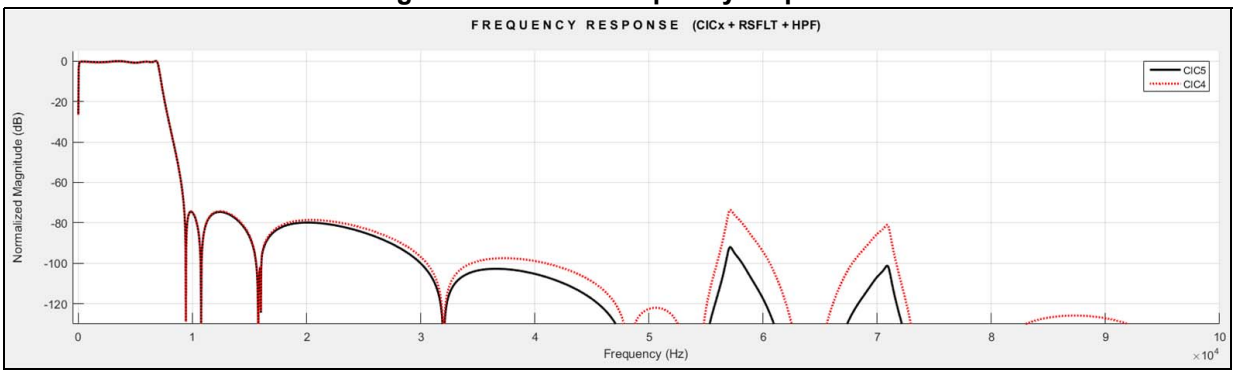


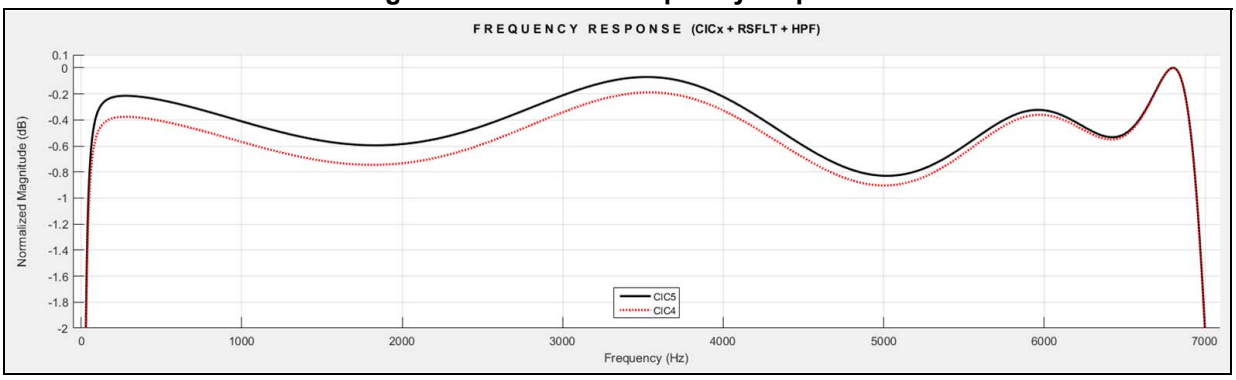
Figure 412 shows the in-band ripple for a 16 kHz audio signal with a digital microphone working at 1.024 MHz. The filter configuration is the following:

- CIC order 4 or 5, with a decimation ratio of 16
- RSFLT enabled, with a decimation ratio of 4
- HPF enabled with a cut-off frequency of 20 Hz

The resulting in-band ripple is ± 0.41 dB for CIC5, and ± 0.45 dB for CIC4.

The - 3 dB cut-off frequency is 7061 Hz.

Figure 412. Detailed frequency response



30.7.5 Total ADF gain

This section details how to compute the signal level provided by the ADF according to the filter settings.

A signal level may be expressed in dBFS (decibel full scale). A 0 dBFS level is assigned to the maximum possible digital level. For example, a signal that reaches 50 % of the maximum level, has a – 6 dBFS level (6 dB below full scale).

For example, for the ADF offering a final data width of 24 bits, a signal having an amplitude of 2 * 10^6 LSB has a level of:

$$20 \times \log_{10}\left(\frac{2 \times 10^6}{2^{(24-1)}}\right) = -12.45 \text{ dBFS}$$

In addition, the data size of a signal having an amplitude (Amp) expressed in LSB is given by:

$$DS = \left(\frac{\ln(Amp)}{\ln(2)} + 1\right) \text{ bits}$$

One bit need to be added for negative values.

So a signal having an amplitude of 2 * 10^6 LSB, has a data size of 21.9 bits.

CIC gain

The CIC gain (G_{CIC} and GdB_{CIC}) can be deduced from the following formula giving data size in bits (DS_{CIC}).

$$DS_{CIC} = (N \times \log_2(D1)) + DSin$$

where N represents the CIC order (selected by CICMOD[2:0]), and D1 is the decimation ratio (given by MCICD[8:0]).

DSin represents the data size (in bits) of the signal at CIC input.

Warning: DS_{CIC} is very important for CIC filter. In order to work fine, DS_{CIC} must not exceed 26 bits.

The CIC gain G_{CIC} is given by:

$$G_{CIC} = (D1)^N$$

which gives in decibels:

$$GdB_{CIC} = 20 \times \log_{10}((D1)^N)$$

Data size at SCALE output

The data size at SCALE output (including the CIC gain), is a key information as the RSFLT starts to have some saturations, if the peak-to-peak signal amplitude at SCALE output is higher than 22 bits.

If the RSFLT is bypassed, then a peak-to-peak signal amplitude of 24 bits is accepted.

The signal amplitude at SCALE output is:

$$Asout_{SCALE} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times Asin_{DFLT}$$

GdB_{SCALE} represents the gain selected by SCALE[5:0], in dB.

$Asout_{SCALE}$ is the signal amplitude at SCALE output (in LSB), and $Asin_{DFLT}$ is the signal amplitude at CIC input (LSB).

$$DS_{SCALE} = \frac{\ln(Asout_{SCALE})}{\ln(2)} + 1$$

The data size at SCALE output (DS_{SCALE}) is expressed in bits.

RSFLT gain

The RSFLT gain in the useful bandwidth is typically 9.5 dB, but due to ripple a margin of about ± 0.41 dB must be considered.

$$G_{RSFLT} = 10^{\frac{9.5\ dB}{20}} = 2.98 \text{ typical}$$

Note: The HPF filter has a gain of 0 dB.

SAD gain

The SAD is using only the 16 MSB on the signal, as a consequence, from the SAD point of view, the truncation from 24 to 16 bits can be seen as an attenuation.

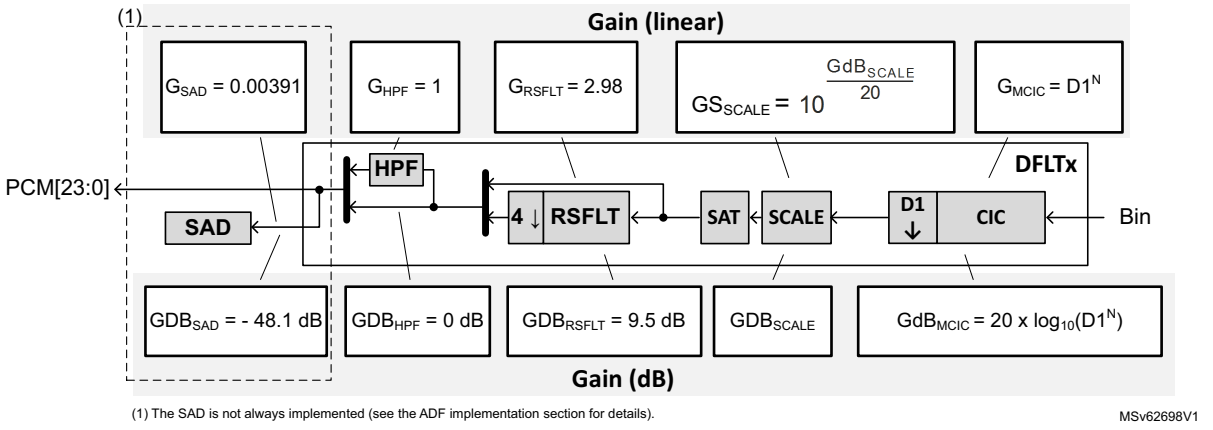
$$GdB_{SAD} = -48.1dB$$

and

$$G_{SAD} = 0.003906$$

The *Figure 413* shows a simplified view of the filter path and gives, for each significant component, the expression of the bit growth and the gain.

Figure 413. Simplified DFLT view with gain information



Note: The SAD is not always implemented (see Section 30.3 for details).

Table 311 summarizes of the final data size for different filter configurations.

Table 311. Output signal levels

Filter configurations	Final signal amplitude (LSB)
CIC + RSFLT + HPF + SAD	Samples provided to the RXFIFO: $Asout_{RXFIFO} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times 10^{\frac{9.5}{20}} \times Asin_{DFLT}$ Samples provided to the SAD: $Asout_{SAD} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times 10^{\frac{9.5}{20}} \times 0.003906 \times Asin_{DFLT}$ DS_{SCALE} must be lower than 22 bits.
CIC + RSFLT (+ HPF)	$ASout_{HPF} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times 10^{\frac{9.5}{20}} \times Asin_{DFLT}$ DS_{SCALE} must be lower than 22 bits.
CIC (+ HPF)	$ASout_{HPF} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times Asin_{DFLT}$ DS_{SCALE} must be lower than 24 bits.

Example using the main filter chain

If the ADF filter is programmed as follows:

- The input signal is coming from a serial interface ($Dsin_{RSFLT}$ = 1 bit).
- CIC order = 5 (N), with a decimation value of 24 (D1).

- SCALE[5:0] is set to - 12 dB.
- RSFLT enabled, and the decimation by four is enabled.
- HPF is enabled.

Check first the data size at CIC output:

$$DS_{CIC} = (5 \times \log_2(24)) + 1 \text{ bit} = 23.92 \text{ bits}$$

The size is lower than 26 bits, so the CIC works in good conditions.

The data size at CIC output is very close to 24 bits, so the SCALE must be adjusted in order to provide a 22-bit max signal to the RSFLT. An attenuation of 12 dB is needed.

Then the signal level provided to the RSFLT is:

$$Asout_{SCALE} = 24^5 \times 10^{\frac{-12}{20}} \times 1 = 2.10^6$$

$$DS_{SCALE} = \frac{\ln(2.10^6)}{\ln(2)} + 1 = 21.93 \text{ bits}$$

If a higher gain is used, the RSFLT may saturate the output signal for strong input signals.

At the end, the final signal amplitude is:

$$Asout_{HPF} = 24^5 \times 10^{\frac{-12}{20}} \times 10^{\frac{9.5}{20}} \times 1 = 5.9711 \times 10^6$$

$$Dsout_{HPF} = \left(\frac{\ln(5.9711 \times 10^6)}{\ln(2)} + 1\right) = 23.51 \text{ bits}$$

or:

$$SDB_{OUT} = 20 \times \log_{10}\left(\frac{2^{23.51}}{2^{24}}\right) = -2.84 \text{ dBFS}$$

30.7.6 How to compute SAD thresholds

The SAD does not compute the RMS value of the converted signal, but the average of the absolute values. As a consequence, the estimated level differs from the RMS value of the signal:

- For a sine signal having an RMS value of 1, the SAD computes a level of 0.9.
- For a white or pink noise signal having an RMS value of 1, the SAD computes a level of about 0.8.

Note: FRSIZE[2:0] has a big influence on the accuracy of the level estimation: big FRSIZE[2:0] values give better results.

Threshold programming with SADMOD = 01

Consider the case of a sound capture where the application wants to wake up the system when the captured sound is bigger than 63 dBSPL.

The sound capture can be performed with a digital microphone such as the MP45DT02.

The sensitivity of this microphone is typically -26 dBFS for an input signal of 94 dBSPL.

An acoustic signal at 63 dBSPL produces a digital signal of about:
- 26 dBFS - (94 - 63) = - 57 dBFS.

- SCALE value adjustment

  For this example, the filter configuration is the following:

  – CIC5 with a decimation by 16
  – RSFLT enabled with a decimation by 4
  – HPF enabled

  A SCALE value of 3.5 dB is recommended for this configuration. As DFTL0 provides samples only used for a sound detection (samples not provided to the application), a bigger gain value can be applied: it increases the SAD accuracy and a saturation does not affect the SAD behavior (for example, a SCALE value of 15.6 dB).

- Input signal amplitude

  The input signal is - 57 dBFS, corresponding to an amplitude

  Asin = $10^{(-57/20)}$ = 0.00141 LSB.

- Signal level at SAD input

  The total filter gain for the SAD is:

  $$G_{SAD} = 16^5 \times 10^{\frac{15.6}{20}} \times 10^{\frac{9.5}{20}} \times \frac{1}{256} = 73.68 \times 10^3 \text{ or } 97.3 \text{ dB}$$

  The signal amplitude received by the SAD is:

  $$Ain_{SAD} = 0.00141 \times G_{SAD} \sim 104 \text{ LSB}$$

  The gain can be increased if the expected amplitude is too small. For the targeted application, 104 LSB is fine.

  If the input signal is expected to be a sine, the sound level for a signal amplitude of 104 LSB is:

  $$SDLVL = \frac{Ain_{SAD} \times \sqrt{2}}{2} \times 0.9 \sim 66 \text{ LSB}$$

  where 0.9 is the correction factor to apply with respect to the RMS value.

- Program the trigger value

  ANMIN and SNTHR must be programmed to trigger the SAD when the input signal level reaches 66 LSB.

  For SADMOD[1:0] = 01, the threshold value is given by:

  $$THRH = ANMIN \times 10^{\frac{GdB_{SNTHR}}{20}}$$

  where GdB_{SNTHR} represents the decibel value selected by SNTHR[3:0].

  When SNTHR[3:0] = 6 dB for example, this formula becomes:

  $$THRH = 2 \times ANMIN$$

  So ANMIN = THRH/2 = 66 /2 = 33 LSB.

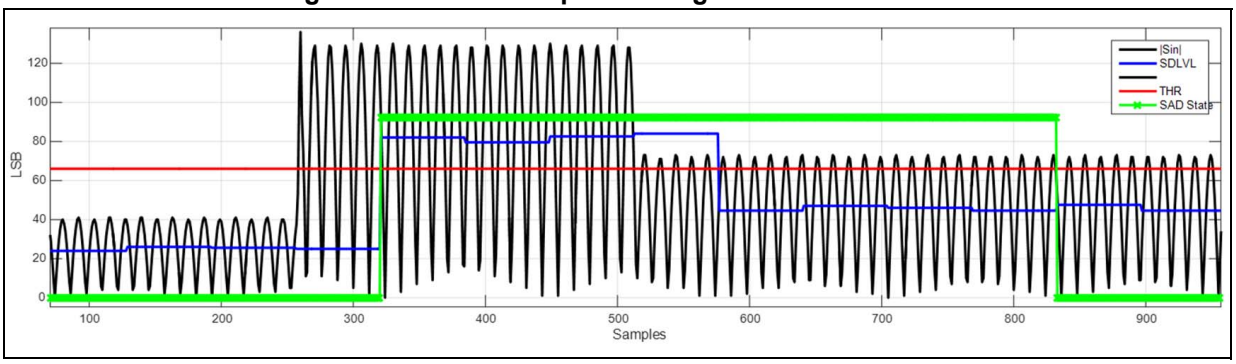
  In *Figure 413*, the trigger value (THRH in red) is fixed to 66 LSB. The input signal is at - 65dBFS during 256 samples, then its value goes to - 55 dB for 256 samples, and finally it is reduced to - 60 dBFS.

The blue curve is showing the sound level estimation (SDLVL) versus time. Fluctuation on the estimated value can be observed due to windowing effect of FRSIZE samples.

The SAD DETECT state (when green signal is high) is maintained during four additional frames due to hangover function value.

In this example ANSLP = FRSIZE = 3 (64 samples), LFRNB = 0 (2 frames), HGOVR = 0 (4 frames), SNTHR = 1 (6 dB) and ANMIN = 33.

Figure 414. SAD example working with SADMOD = 01



Threshold programming with SADMOD = 1x

Consider the case of a sound capture where the application wants to wake up the system when the captured sound is bigger than 57 dBSPL.

The sound capture can be performed with a digital microphone such as the MP45DT02. The sensitivity of this microphone is typically - 26 dBFS for an input signal of 94 dBSPL.

An acoustic signal at 57 dBSPL produces a digital signal of about:
- 26 dBFS - (94-63) = - 63 dBFS.

- Adjust SCALE value

  For this example, the filter configuration is the following:

  – CIC4 with a decimation by 48
  – RSFLT bypassed
  – HPF enabled

  A SCALE value of 3.5 dB is recommended for this configuration. The samples provided by DFTL0 are only used for a sound detection, without providing the samples to the application, a bigger gain value can be provided: it increases the SAD accuracy and a saturation does not affect the SAD behavior (for example, a SCALE value of 24 dB).

- Input signal amplitude

  The input signal is - 63 dBFS, corresponding to an amplitude:

  Asin = $10^{(-63/20)}$ = 0.000708 LSB.

- Signal level at SAD input

  The total filter gain for the SAD is:

$$G_{SAD} = 48^4 \times 10^{\frac{24}{20}} \times 0.003906 = 328.6 \times 10^3 \text{ or } 110.3 \text{ dB}$$

  The signal amplitude received by the SAD is:

$$Ain_{SAD} = 0.000708 \times G_{SAD} \sim 232 \text{ LSB}$$

The gain can be increased if the expected amplitude is too small.

If the input signal is expected to be a sine, the sound level for a signal amplitude of 232 LSB is:

$$SDLVL = \frac{Ain_{SAD} \times \sqrt{2}}{2} \times 0.9 \sim 148 \text{ LSB}$$

where 0.9 is the correction factor to apply with respect to the RMS value.

Note: ANLVL converges to average of SDLVL values, with a long constant time.
So SDLVL ~ ANLVL = 148 LSB for a constant input signal at 57 dBSPL.

- Programming trigger value

  For SADMOD = 1', the SAD compares the estimated ambient noise multiplied by the gain selected by SNTHR[3:0] to ANMIN[12:0] * 4.

  For simplification, SNTHR[3:0] is set to 1 (6 dB), meaning that ANLVL is multiplied by two.

  The SAD triggers if 2 * ANLVL > THRH.

  In this mode:

  $$THRH = 4 \times ANMIN$$

  So the SAD triggers if:

  $$ANLVL > 2 \times ANMIN$$

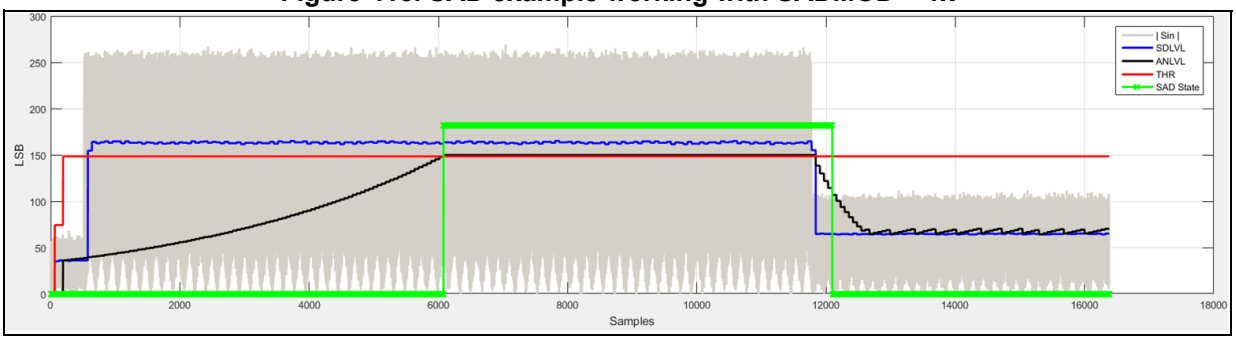
  So ANMIN = 148 / 2 = 74 LSB

  In *Figure 414*, the trigger value (THRH in red) is fixed to 148 LSB. The input signal is at - 75 dBFS during 512 samples, then its value goes to - 62 dB for 11000 samples, and finally it is reduced to - 70 dBFS.

  The blue curve shows the sound level estimation (SDLVL) versus time. The black curve shows the ambient noise estimation versus time, increasing or decreasing logarithmically. During the learning phase, it reaches the SDLVL value.

  In this example ANSLP = 6, FRSIZE = 3 (64 samples), LFRNB = 0 (2 frames), HGOVR = 0 (4 frames), SNTHR = 1 (6 dB) and ANMIN = 74.

Figure 415. SAD example working with SADMOD = 1x



30.8 ADF registers

All the ADF registers must be accessed either in word (32-bit) or half-word (16-bit) formats.

30.8.1 ADF global control register (ADF_GCR)

Address offset: 0x000

Reset value: 0x0000 0000

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

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

Bit 0 **TRGO**: Trigger output control

This bit is set by software and reset by hardware. It is used to start the acquisition of several filters synchronously. It is also used to synchronize several ADF together by controlling the adf_trgo signal.

0: Write 0 has no effect. Read 0 means that the trigger can be set again to 1.

1: Write 1 generates a positive pulse on the adf_trgo signal and triggers the acquisition on enabled filter having their ACQMOD[2:0] = 01x and selecting TRGO as trigger. Read 1 means that the trigger pulse is still active.

30.8.2 ADF clock generator control register (ADF_CKGCR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is used to control the clock generator. The clock adf_proc_ck must be enabled before enabling other ADF parts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CKGACTIVE	PROCDIV[6:0]							Res.	Res.	Res.	Res.	CCKDIV[3:0]			
r	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	CCK1DIR	CCK0DIR	CKGMOD	Res.	CCK1EN	CCK0EN	CKGDEN
rw	rw	rw	rw				rw		rw	rw	rw		rw	rw	rw

Bit 31 **CKGACTIVE**: Clock generator active flag

This bit is set and cleared by hardware. Ii is used by the application to check if the clock generator is effectively enabled (active) or not. The protected fields of this function can only be updated when CKGACTIVE = 0 (see *Section 30.4.13: Register protection* for details).

The delay between a transition on CKGDEN and a transition on CKGACTIVE is two periods of AHB clock and two 2 periods of adf_proc_ck.

0: The clock generator is not active and can be configured if needed.

1: The clock generator is active and protected fields cannot be configured.

Bits 30:24 **PROCDIV[6:0]**: Divider to control the serial interface clock

This bitfield is set and reset by software. It is used to adjust the frequency of the clock provided to the SITF.

$$F_{adf_itf_ck} = \frac{F_{adf_ker_ck}}{(PROCDIV + 1)}$$

This bitfield must not be changed if the filter is enabled (DFTEN = 1).

0: adf_ker_ck provided to the SITF

1: adf_ker_ck / 2 provided to the SITF

2: adf_ker_ck / 3 provided to the SITF

...

127: adf_ker_ck / 128 provided to the SITF

Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

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

Bits 19:16 **CCKDIV[3:0]**: Divider to control the ADF_CCK clock

This bitfield is set and reset by software. It is used to adjust the frequency of the ADF_CCK clock. The input clock of this divider is the clock provided to the SITF. More globally, the frequency of the ADF_CCK is given by the following formula:

$$F_{ADF_CCK} = \frac{F_{adf_ker_ck}}{(PROCDIV + 1) \times (CCKDIV + 1)}$$

This bitfield must not be changed if the filter is enabled (DFTEN = 1).

0000: The ADF_CCK clock is adf_proc_ck.

0001: The ADF_CCK clock is adf_proc_ck / 2.

0010: The ADF_CCK clock is adf_proc_ck / 3.

...

1111: The ADF_CCK clock is adf_proc_ck / 16.

Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bits 15:12 **TRGSRC[3:0]**: Digital filter trigger signal selection

This bitfield is set and cleared by software. It is used to select which external signals trigger the corresponding filter. This bitfield is not significant if the CKGMOD = 0.

000x: TRGO selected

0010: adf_trg1 selected

Others: Reserved

Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGSENS**: CKGEN trigger sensitivity selection

This bit is set and cleared by software. It is used to select the trigger sensitivity of the trigger signals. This bit is not significant if the CKGMOD = 0.

0: A rising edge event triggers the activation of CKGEN dividers.

1: A falling edge even triggers the activation of CKGEN dividers.

Note: When the trigger source is TRGO, the sensitivity is forced to falling edge, thus TRGSENS value is not taken into account. This bit can be write-protected (see Section 30.4.13: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CCK1DIR**: ADF_CCK1 direction

This bit is set and reset by software. It is used to control the direction of the ADF_CCK1 pin.

0: The ADF_CCK1 pin direction is in input.

1: The ADF_CCK1 pin direction is in output.

Note: This bit can be write-protected (see Section 30.4.13: Register protection for details).

Bit 5 **CCK0DIR**: ADF_CCK0 direction

This bit is set and reset by software. It is used to control the direction of the ADF_CCK0 pin.

0: The ADF_CCK0 pin direction is in input.

1: The ADF_CCK0 pin direction is in output.

Note: This bit can be write-protected (see Section 30.4.13: Register protection for details).

Bit 4 **CKGMOD**: Clock generator mode

This bit is set and reset by software. It is used to define the way the clock generator is enabled. This bit must not be changed if the filter is enabled (DFTEN = 1).

0: The kernel clock is provided to the dividers as soon as CKGDEN is set to 1.

1: The kernel clock is provided to the dividers when CKGDEN is set to 1 and the trigger condition met.

Note: This bit can be write-protected (see Section 30.4.13: Register protection for details).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CCK1EN**: ADF_CCK1 clock enable

This bit is set and reset by software. It is used to control the generation of the bitstream clock on the ADF_CCK1 pin.

0: Bitstream clock not generated

1: Bitstream clock generated on the ADF_CCK1 pin.

Bit 1 **CCK0EN**: ADF_CCK0 clock enable

This bit is set and reset by software. It is used to control the generation of the bitstream clock on the ADF_CCK0 pin.

0: Bitstream clock not generated

1: Bitstream clock generated on the ADF_CCK0 pin

Bit 0 **CKGDEN**: CKGEN dividers enable

This bit is set and reset by software. It is used to enable/disable the clock dividers of the CKGEN: PROCDIV and CCKDIV.

0: CKGEN dividers disabled

1: CKGEN dividers enabled

30.8.3 ADF serial interface control register 0 (ADF_SITF0CR)

Address offset: 0x080

Reset value: 0x0000 1F00

This register is used to control the serial interface SITF0.

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

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

Bit 31 **SITFACTIVE**: Serial interface active flag

This bit is set and cleared by hardware. It is used by the application to check if the serial interface is effectively enabled (active) or not. The protected fields of this function can only be updated when SITFACTIVE is set to 0 (see *Section 30.4.13: Register protection* for details).
The delay between a transition on SITFEN and a transition on SITFACTIVE is two periods of AHB clock and two periods of adf_proc_ck.
0: The serial interface is not active, and can be configured if needed.
1: The serial interface is active and protected fields cannot be configured.

Bits 30:13 Reserved, must be kept at reset value.

Bits 12:8 **STH[4:0]**: Manchester symbol threshold/SPI threshold

This bitfield is set and cleared by software. It is used for Manchester mode to define the expected symbol threshold levels (see *Manchester mode* for details on computation).
In addition this bitfield is used to define the timeout value for the clock absence detection in Normal SPI mode. STH[4:0] values lower than four are invalid.

Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **SITFMOD[1:0]**: Serial interface type

This bitfield is set and cleared by software. It is used to define the serial interface type.
00: LF_MASTER SPI mode
01: Normal SPI mode
10: Manchester mode: rising edge = logic 0, falling edge = logic 1
11: Manchester mode: rising edge = logic 1, falling edge = logic 0

Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **SCKSRC[1:0]**: Serial clock source

This bitfield is set and cleared by software. It is used to select the clock source of the serial interface.
00: Serial clock source is ADF_CCK0.
01: Serial clock source is ADF_CCK1.
others: reserved

Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bit 0 **SITFEN**: Serial interface enable

This bit is set and cleared by software. It is used to enable/disable the serial interface.
0: Serial interface disabled
1: Serial interface enabled

30.8.4 ADF bitstream matrix control register 0 (ADF_BSMX0CR)

Address offset: 0x084

Reset value: 0x0000 0000

This register is used to select the bitstream to be provided to DFLT0.

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

Bit 31 **BSMXACTIVE**: BSMX active flag

This bit is set and cleared by hardware. It is used by the application to check if the BSMX is effectively enabled (active) or not. BSSEL[4:0] can only be updated when BSMXACTIVE is set to 0. This BSMXACTIVE flag cannot go to 0 if DFLT0 is enabled.

0: BSMX is not active and can be configured if needed.

1: BSMX is active and protected fields cannot be configured.

Bits 30:5 Reserved, must be kept at reset value.

Bits 4:0 **BSSEL[4:0]**: Bitstream selection

This bitfield is set and cleared by software. It is used to select the bitstream to be processed for DFLT0.

00000: bs0_r provided to DFLT0

00001: bs0_f provided to DFLT0

others: reserved

Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

30.8.5 ADF digital filter control register 0 (ADF_DFLT0CR)

Address offset: 0x088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DFLTACTIVE	DFLTRUN	Res.	Res.	NBDIS[7:0]								Res.	Res.	Res.	Res.
r	r			rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	ACQMOD[2:0]			Res.	FTH	DMAEN	DFLTEN
rw	rw	rw	rw				rw		rw	rw	rw		rw	rw	w

Bit 31 **DFLTACTIVE**: DFLT0 active flag

This bit is set and cleared by hardware. It indicates if DFLT0 is active: can be running or waiting for events.

0: DFLT0 not active (can be re-enabled again, via DFLTEN bit, if needed)

1: DFLT0 active

Bit 30 **DFLTRUN**: DFLT0 run status flag
This bit is set and cleared by hardware. It indicates if DFLT0 is running or not.
0: DFLT0 not running and ready to accept a new trigger event
1: DFLT0running

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:20 **NBDIS[7:0]**: Number of samples to be discarded
This bitfield is set and cleared by software. It is used to define the number of samples to be discarded every time DFLT0 is re-started.
0: No sample discarded
1: 1 sample discarded
2: 2 samples discarded
...
255: 255 samples discarded
Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:12 **TRGSRC[3:0]**: DFLT0 trigger signal selection
This bitfield is set and cleared by software. It is used to select which external signals trigger DFLT0.
0000: TRGO selected
0010: adf_trgi selected
Others: Reserved
Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGSENS**: DFLT0 trigger sensitivity selection
This bitfield is set and cleared by software. It is used to select the trigger sensitivity of the external signals
0: A rising edge event triggers the acquisition.
1: A falling edge even triggers the acquisition.
Note: When the trigger source is TRGO, TRGSENS value is not taken into account. When TRGO is selected, the sensitivity is forced to falling edge.
This bit can be write-protected (see Section 30.4.13: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ACQMOD[2:0]**: DFLT0 trigger mode
This bitfield is set and cleared by software. It is used to select the filter trigger mode.
000: Asynchronous continuous acquisition mode
001: Asynchronous single-shot acquisition mode
010: Synchronous continuous acquisition mode
011: Synchronous single-shot acquisition mode
100: Window continuous acquisition mode
Others: Same as 000
Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FTH**: RXFIFO threshold selection
This bit is set and cleared by software. It is used to select the RXFIFO threshold.
0: RXFIFO threshold event generated when the RXFIFO is not empty
1: RXFIFO threshold event generated when the RXFIFO is half-full
Note: This bit can be write-protected (see Section 30.4.13: Register protection for details).

Bit 1 **DMAEN**: DMA requests enable

This bit is set and cleared by software. It is used to control the generation of DMA request to transfer the processed samples into the memory.

0: DMA interface for the corresponding digital filter disabled

1: DMA interface for the corresponding digital filter enabled

Note: This bit can be write-protected (see Section 30.4.13: Register protection for details).

Bit 0 **DFLTEN**: DFLT0 enable

This bit is set and cleared by software. It is used to control the start of acquisition of the DFLT0 path. This bit behavior depends on ACQMOD[2:0] and external events. The serial or parallel interface delivering the samples must be enabled as well.

0: Acquisition immediately stopped

1: Acquisition immediately started if ACQMOD[2:0] = 00x or 101, or acquisition started when the proper trigger event occurs if ACQMOD[2:0] = 01x.

30.8.6 ADF digital filer configuration register 0 (ADF_DFLT0CICR)

Address offset: 0x08C

Reset value: 0x0000 0000

This register is used to control the main CIC filter.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SCALE[5:0]						Res.	Res.	Res.	MCICD 8
						rw	rw	rw	rw	rw	rw				rw

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

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

Bits 25:20 **SCALE[5:0]**: Scaling factor selection

This bitfield is set and cleared by software. It is used to select the gain to be applied at CIC output (see *Table 299* for details). If the application attempts to write a new gain value while the previous one is not yet applied, this new gain value is ignored. Reading back this bitfield informs the application on the current gain value.

000000: 0 dB

000001: + 3.5 dB

000010: + 6 dB or shift left by 1 bit

...

011000: + 72 dB or shift left by 12 bits

100000: - 48.2 dB or shift right by 8 bits (default value)

100001: - 44.6 dB

100010: - 42.1 dB or shift right by 7 bits

100011: - 38.6 dB

...

101110: -6 dB or shift right by 1 bit

101111: -2.5 dB

Others: Reserved

Bits 19:17 Reserved, must be kept at reset value.

Bits 16:8 **MCICD[8:0]**: CIC decimation ratio selection
This bitfield is set and cleared by software.It is used to select the CIC decimation ratio. A decimation ratio smaller than two is not allowed. The decimation ratio is given by (CICDEC+1).
0: Decimation ratio is 2.
1: Decimation ratio is 2.
2: Decimation ratio is 3.
3: Decimation ratio is 4.
...
511: Decimation ratio is 512.
Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **CICMOD[2:0]**: Select the CIC order
This bitfield is set and cleared by software. It is used to select the order of the MCIC.
100: MCIC configured in single Sinc^4 filter
101: MCIC configured in single Sinc^5 filter
Others: Reserved
Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **DATSRC[1:0]**: Source data for the digital filter
This bitfield is set and cleared by software.
0x: Stream coming from the BSMX selected
10: Stream coming from the ADCITF1 selected
11: Stream coming from the ADCITF2 selected
Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

30.8.7 ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)

Address offset: 0x090

Reset value: 0x0000 0000

This register is used to control the reshape and HPF filter.

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

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

Bits 9:8 **HPFC[1:0]**: High-pass filter cut-off frequency
This bitfield is set and cleared by software. it is used to select the cut-off frequency of the high-pass filter. F_{PCM} represents the sampling frequency at HPF input.
00: Cut-off frequency = 0.000625 x FPCM
01: Cut-off frequency = 0.00125 x FPCM
10: Cut-off frequency = 0.00250 x FPCM
11: Cut-off frequency = 0.00950 x FPCM
Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bit 7 **HPFBYP**: High-pass filter bypass
This bit is set and cleared by software. It is used to bypass the high-pass filter.
0: HPF not bypassed (default value)
1: HPF bypassed
Note: This bit can be write-protected (see Section 30.4.13: Register protection for details).

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **RSFLTD**: Reshaper filter decimation ratio
This bit is set and cleared by software. It is used to select the decimation ratio for the reshape filter
0: Decimation ratio is 4 (default value).
1: Decimation ratio is 1.
Note: This bit can be write-protected (see Section 30.4.13: Register protection for details).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **RSFLTBYP**: Reshaper filter bypass
This bit is set and cleared by software. It is used to bypass the reshape filter and its decimation block.
0: Reshape filter not bypassed (default value)
1: Reshape filter bypassed
Note: This bit can be write-protected (see Section 30.4.13: Register protection for details).

30.8.8 ADF delay control register 0 (ADF_DLY0CR)

Address offset: 0x0A4

Reset value: 0x0000 0000

This register is used for the adjustment stream delays.

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

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

Bit 31 **SKPBF**: Skip busy flag
This bit is set and cleared by hardware. It is used to control if the delay sequence is completed.
0: ADF ready to accept a new value into SKPDLY[6:0]
1: Last valid SKPDLY[6:0] still under precessing

Bits 30:7 Reserved, must be kept at reset value.

Bits 6:0 **SKPDLY[6:0]**: Delay to apply to a bitstream

This bitfield is set and cleared by software. It defines the number of input samples that are skipped. Skipping is applied immediately after writing to this bitfield, if SKPBF = 0 and DFLTEN = 1. If SKPBF = 1, the value written into the register is ignored by the delay state machine.

0: No input sample skipped

1: 1 input sample skipped

...

127: 127 input samples skipped

30.8.9 ADF DFLT0 interrupt enable register (ADF_DFLT0IER)

Address offset: 0x0AC

Reset value: 0x0000 0000

This register is used for allowing or not the events to generate an interrupt.

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

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **SDLVLIE**: SAD sound-level value ready enable

This bit is set and cleared by software.

0: Sound-level-ready interrupt disabled

1: Sound-level-ready interrupt enabled

Bit 12 **SDDETIE**: Sound activity detection interrupt enable

This bit is set and cleared by software.

0: Sound-trigger interrupt disabled

1: Sound-trigger interrupt enabled

Bit 11 **RFOVRIE**: Reshape filter overrun interrupt enable

This bit is set and cleared by software.

0: Reshape filter overrun interrupt disabled

1: Reshape filter overrun interrupt enabled

Bit 10 **CKABIE**: Clock absence detection interrupt enable

This bit is set and cleared by software.

0: Clock absence interrupt disabled

1: Clock absence interrupt enabled

Bit 9 **SATIE**: Saturation detection interrupt enable

This bit is set and cleared by software.

0: Saturation interrupt disabled

1: Saturation interrupt enabled

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **DOVRIE**: Data overflow interrupt enable
This bit is set and cleared by software.
0: Data overflow interrupt disabled
1: Data overflow interrupt enabled

Bit 0 **FTHIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: RXFIFO threshold interrupt disabled
1: RXFIFO threshold interrupt enabled

30.8.10 ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)

Address offset: 0x0B0

Reset value: 0x0000 0000

This register contains the status flags for the digital filter path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SDLVLF	SDDETF	RFOVRF	CKABF	SATF	Res.	Res.	Res.	Res.	Res.	RXNEF	Res.	DOVRF	FTHF
		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1						r		rc_w1	r

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **SDLVLF**: Sound level value ready flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that new sound level value is not ready. Write 0 has no effect.
1: Read 1 means that new sound level value is ready. Write 1 clears this flag.

Bit 12 **SDDETF**: Sound activity detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no sound activity is detected. Write 0 has no effect.
1: Read 1 means that sound activity is detected. Write 1 clears this flag.

Bit 11 **RFOVRF**: Reshape filter overrun detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no reshape filter overrun is detected. Write 0 has no effect.
1: Read 1 means that reshape filter overrun is detected. Write 1 clears this flag.

Bit 10 **CKABF**: Clock absence detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no clock absence is detected. Write 0 has no effect.
1: Read 1 means that a clock absence is detected. Write 1 clears this flag.

Bit 9 **SATF**: Saturation detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no saturation is detected. Write 0 has no effect.
1: Read 1 means that a saturation is detected. Write 1 clears this flag.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **RXNEF**: RXFIFO not empty flag
This bit is set and cleared by hardware according to the RXFIFO level.
0: RXFIFO empty
1: RXFIFO not empty

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DOVRF**: Data overflow flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no overflow is detected. Write 0 has no effect.
1: Read 1 means that an overflow is detected; Write 1 clears this flag.

Bit 0 **FTHF**: RXFIFO threshold flag
This bit is set by hardware, and cleared by the hardware when the RXFIFO level is lower than the threshold.
0: RXFIFO threshold not reached
1: RXFIFO threshold reached

30.8.11 ADF SAD control register (ADF_SADCR)

Address offset: 0x0B8

Reset value: 0x0000 0000

This register is used for the configuration and the control of the sound activity detection.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SADACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SADMOD[1:0]		Res.	FRSIZE[2:0]			HYSTEN	Res.	SADST[1:0]		DETCFG	DATCAP[1:0]		SADEN
		rw	rw		rw	rw	rw	rw		r	r	rw	rw	rw	rw

Bit 31 **SADACTIVE**: SAD Active flag
This bit is set and cleared by hardware. It is used to check if the SAD is effectively enabled (active) or not. The protected fields and registers of this function can only be updated when the SADACTIVE is set to 0 (see *Section 30.4.13: Register protection* for details).
The delay between a transition on SADEN and a transition on SADACTIVE is two periods of AHB clock and two periods of adf_proc_ck.
0: SAD not active and can be configured if needed
1: SAD active and protected fields cannot be configured.

Bits 30:14 Reserved, must be kept at reset value.

Bits 13:12 **SADMOD[1:0]**: SAD working mode

This bitfield is set and cleared by software. It is used to define the way the SAD works.

00: Threshold value computed according to the estimated ambient noise

The SAD triggers when the sound level (SDLVL) is bigger than the defined threshold. In this mode, the SAD works like a voice activity detector.

01: Threshold value equal to ANMIN[12:0], multiplied by the gain selected by SNTHR[3:0]

The SAD triggers when the sound level (SDLVL) is bigger than the defined threshold. In this mode, the SAD works like a sound detector.

1x: Threshold value given by 4 x ANMIN[12:0]

The SAD triggers when the estimated ambient noise (ANLVL), multiplied by the gain selected by SNTHR[3:0] is bigger than the defined threshold. In this mode, the SAD is working like an ambient noise estimator. Hysteresis function cannot be used in this mode.

Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **FRSIZE[2:0]**: Frame size

This bitfield is set and cleared by software. it is used to define the size of one frame and also to define how many samples are taken into account to compute the short-term signal level.

000: 8 PCM samples used to compute the short-term signal level

001: 16 PCM samples used to compute the short-term signal level

010: 32 PCM samples used to compute the short-term signal level

011: 64 PCM samples used to compute the short-term signal level

100: 128 PCM samples used to compute the short-term signal level

101: 256 PCM samples used to compute the short-term signal level

11x: 512 PCM samples used to compute the short-term signal level

Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bit 7 **HYSTEN**: Hysteresis enable

This bit is set and cleared by software. It is used to enable/disable the hysteresis function (see *Table 299* for details). This bit must be kept to 0 when SADMOD[1:0] = 1x.

0: Hysteresis function disabled. THR_H is always used.

1: Hysteresis function enabled. THR_H is used for MONITOR to DETECT transition and THR_L is used for DETECT to MONITOR transition.

Note: This bit can be write-protected (see Section 30.4.13: Register protection for details).

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **SADST[1:0]**: SAD state

This bitfield is set and cleared by hardware. It indicates the SAD state and is meaningful only when SADEN = 1. The SAD state can be:

- LEARN when the SAD is in learning phase or in SDLVL computation mode
- MONITOR when the SAD is in monitoring phase
- DETECT when the SAD detects a sound

00: SAD in LEARN state

01: SAD in MONITOR state

11: SAD in DETECT state

Bit 3 **DETCFG**: Sound trigger event configuration

This bit is set and cleared by software. It is used to define if the sddet_evt event is generated only when the SAD enters to MONITOR state or when the SAD enters or exits the DETECT state.

0: sddet_evt generated when SAD enters the MONITOR state

1: sddet_evt generated when SAD enters or exits the DETECT state

Note: This bit can be write-protected (see Section 30.4.13: Register protection for details).

Bits 2:1 **DATCAP[1:0]**: Data capture mode
This bitfield is set and cleared by software. It is used to define in which conditions, the samples provided by DLFT0 are stored into the memory.
00: Samples from DFLT0 not transfered into the memory
01: Samples from DFLT0 transfered into the memory when SAD is in DETECT state
1x: Samples from DFLT0 transfered into memory when SAD and DFLT0 are enabled
Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bit 0 **SADEN**: Sound activity detector enable
This bit is set and cleared by software. It is used to enable/disable the SAD.
0: SAD disabled and SAD state reset
1: SAD enabled

30.8.12 ADF SAD configuration register (ADF_SADCFGR)

Address offset: 0x0BC

Reset value: 0x0000 0000

This register is used for the configuration of the sound activity detection.

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **ANMIN[12:0]**: Minimum noise level
This bitfield is set and cleared by software. It is used to define the minimum noise level and then the sensitivity. It represents a positive number.
Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **HGOVR[2:0]**: Hangover time window
This bitfield is set and cleared by software. Once the SAD state is DETECT, this parameter is used to define the amount of time the sound is allowed to remain below the threshold, before switching the SAD to MONITOR state (see FRSIZE bitfield for the description of a frame).
000: SAD back to MONITOR state if sound is below threshold for 4 frames
001: SAD back to MONITOR state if sound is below threshold for 8 frames
010: SAD back to MONITOR state if sound is below threshold for 16 frames
011: SAD back to MONITOR state if sound is below threshold for 32 frames
100: SAD back to MONITOR state if sound is below threshold for 64 frames
101: SAD back to MONITOR state if sound is below threshold for 128 frames
110: SAD back to MONITOR state if sound is below threshold for 256 frames
111: SAD back to MONITOR state if sound is below threshold for 512 frames
Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **LFRNB[2:0]**: Number of learning frames

This bitfield is set and cleared by software. It is used to define the number of learning frames to perform the first estimate of the noise level.

000: 2 frames used to compute the initial noise level
001: 4 frames used to compute the initial noise level
010: 8 frames used to compute the initial noise level
011: 16 frames used to compute the initial noise level
1xx: 32 frames used to compute the initial noise level

Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ANSLP[2:0]**: Ambient noise slope control

This bitfield is set and cleared by software. It is used to define the positive and negative slope of the noise estimator, in charge of updating the ANLVL (see *Ambient noise estimation (ANLVL)* for information about programming this bitfield).

Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

Bits 3:0 **SNTHR[3:0]**: Signal to noise threshold

This bitfield is set and cleared by software. It is used to define THR_H (and THR_L if hysteresis is enabled). See *Table 299* for details.

0000: THR_H is 3.5 dB higher than ANLVL
0001: THR_H is 6.0 dB higher than ANLVL
0010: THR_H is 9.5 dB higher than ANLVL
0011: THR_H is 12 dB higher than ANLVL
0100: THR_H is 15.6 dB higher than ANLVL
0101: THR_H is 18 dB higher than ANLVL
0110: THR_H is 21.6 dB higher than ANLVL
0111: THR_H is 24.1 dB higher than ANLVL
1000: THR_H is 27.6 dB higher than ANLVL
1001: THR_H is 30.1dB higher than ANLVL
others: Reserved

Note: This bitfield can be write-protected (see Section 30.4.13: Register protection for details).

30.8.13 ADF SAD sound level register (ADF_SADSDLVR)

Address offset: 0x0C0

Reset value: 0x0000 0000

This register contains the short-term sound-level computed by the SAD.

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

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

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **SDLVL[14:0]**: Short term sound level

This bitfield is set by hardware. It contains the latest sound level computed by the SAD. To refresh this value, SDLVLF must be cleared.

30.8.14 ADF SAD ambient noise level register (ADF_SADANLVR)

Address offset: 0x0C4

Reset value: 0x0000 0000

This register contains the ambient noise level computed by the SAD.

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

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **ANLVL[14:0]**: Ambient noise level estimation

This bitfield is set by hardware. It contains the latest ambient noise level computed by the SAD. To refresh this bitfield, the SDLVLF flag must be cleared.

30.8.15 ADF digital filter data register 0 (ADF_DFLT0DR)

Address offset: 0x0F0

Reset value: 0x0000 0000

This register is used to read the data processed by the digital filter.

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

Bits 31:8 **DR[23:0]**: Data processed by DFT0

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

30.8.16 ADF register map

Table 312. ADF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ADF_GCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGO
	Reset value																																0
0x004	ADF_CKGCR	CKGACTIVE	PROCDIV[6:0]							Res.	Res.	Res.	Res.	CCKDIV [3:0]				TRGSRC [3:0]				Res.	Res.	Res.	TRGSENS	Res.	CCK1DIR	CCK0DIR	CKGMOD	Res.	CCK1EN	CCK0EN	CKGDEN
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0				0		0	0	0		0	0	0

Table 312. ADF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x008 - 0x07C	Reserved																																
0x080	ADF_SITF0CR	SITFACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STH[4:0]					Res.	Res.	SITFMOD [1:0]		Res.	SCKSRC [1:0]		SITFEN
	Reset value	0																			1	1	1	1	1			0	0		0	0	0
0x084	ADF_BSMX0CR	BSMXACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSSEL[4:0]				
	Reset value	0																											0	0	0	0	0
0x088	ADF_DFLT0CR	DFLTACTIVE	DFLTRUN	Res.	Res.	NBDIS[7:0]								Res.	Res.	Res.	Res.	TRGSRC [3:0]				Res.	Res.	Res.	TRGSENS	Res.	ACQMOD [2:0]			Res.	FTH	DMAEN	DFLTEN
	Reset value	0	0			0	0	0	0	0	0	0	0					0	0	0	0				0		0	0	0		0	0	0
0x08C	ADF_DFLT0CICR	Res.	Res.	Res.	Res.	Res.	Res.	SCALE[5:0]						Res.	Res.	Res.	MCICD[8:0]									Res.	CICMOD[2:0]			Res.	Res.	DATSRC[1:0]	
	Reset value							0	0	0	0	0	0				0	0	0	0	0	0	0	0	0		0	0	0			0	0
0x090	ADF_ DFLT0RSFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HPFC[1:0]		HPFBYP	Res.	Res.	RSFLTD	Res.	Res.	Res.	RSFLTBYP
	Reset value																							0	0	0			0				0
0x094 - 0x0A0	Reserved																																
0x0A4	ADF_DLY0CR	SKPBF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SKPDLY[6:0]						
	Reset value	0																									0	0	0	0	0	0	0
0x0A8	Reserved																																
0x0AC	ADF_DFLT0IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDLVLIE	SDDETIE	RFOVRIE	CKABIE	SATIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOVRIE	FTHIE
	Reset value																			0	0	0	0	0	0	0			0		0	0	0
0x0B0	ADF_DFLT0ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDLVLF	SDDETF	RFOVRF	CKABF	SATF	Res.	Res.	Res.	Res.	Res.	RXNEF	Res.	DOVRF	FTHF
	Reset value																			0	0	0	0	0						0		0	0
0x0B4	Reserved																																

Table 312. ADF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B8	ADF_SADCR	SADACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SADMOD[1:0]		Res.	FRSIZE[2:0]			HYSTEN	Res.	SADST[1:0]		DETCFG	DATCAP[1:0]		SADEN
	Reset value	0																		0	0		0	0	0	0		0	0	0	0	0	0
0x0BC	ADF_SADCFGR	Res.	Res.	Res.	ANMIN[12:0]													Res.	HGOVR [2:0]			Res.	LFRNB [2:0]			Res.	ANSLP[2:0]			Res.	SNTHR [2:0]		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0		0	0	0		0	0	0		0	0	0
0x0C0	ADF_SADSDLVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDLVL[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C4	ADF_SADANLVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANLVL[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C8 - 0x0EC	Reserved																																
0x0F0	ADF_DFLT0DR	DR[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

31 True random number generator (RNG)

31.1 RNG introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG is an NIST SP 800-90B compliant entropy source that can be used to construct a nondeterministic random bit generator (NDRBG).

The RNG true random number generator can be certified NIST SP 800-90B. It can also be tested using the German BSI statistical tests of AIS-31 (T0 to T8).

31.2 RNG main features

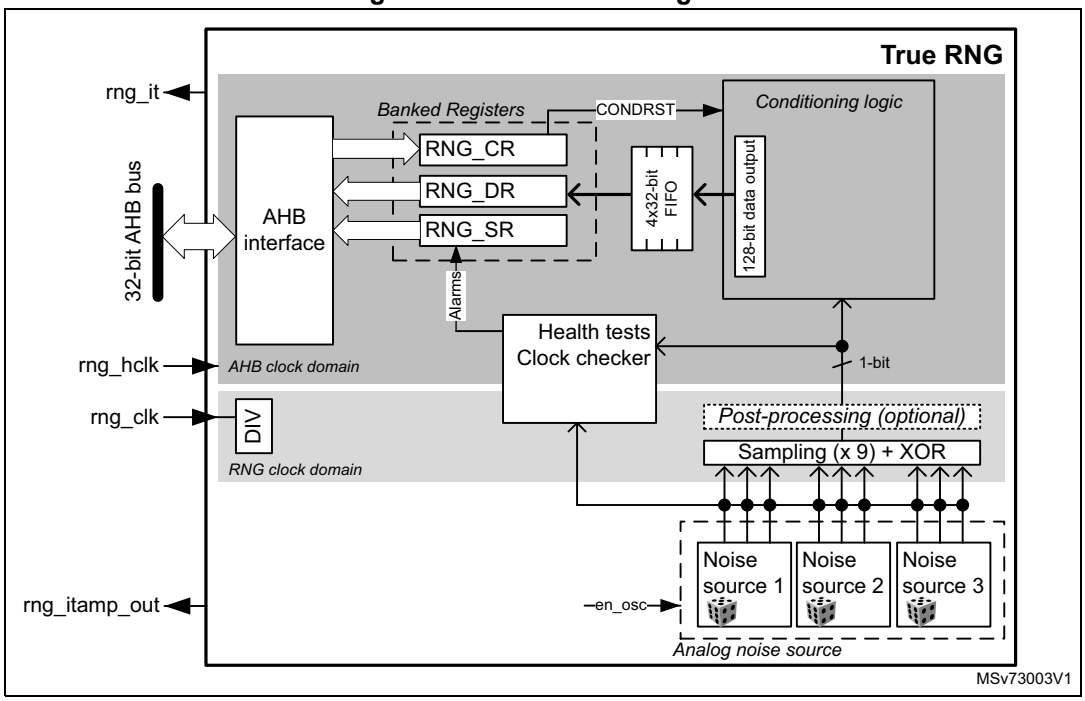
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source conditioned by an NIST SP800-90B approved conditioning stage.
- It can be used as the entropy source to construct a non-deterministic random bit generator (NDRBG).
- In the default configuration, it produces four 32-bit random samples every 412 AHB clock cycles if $f_{AHB} < f_{threshold}$ (256 RNG clock cycles otherwise).
- Embeds startup and NIST SP800-90B approved continuous health tests (repetition count and adaptive proportion tests), associated with specific error management.
- Can be disabled to reduce power consumption, or enabled with an automatic low power mode (default configuration).
- Has an AMBA® AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).

31.3 RNG functional description

31.3.1 RNG block diagram

Figure 416 shows the RNG block diagram.

Figure 416. RNG block diagram



31.3.2 RNG internal signals

Table 313 describes internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 313. RNG internal input/output signals

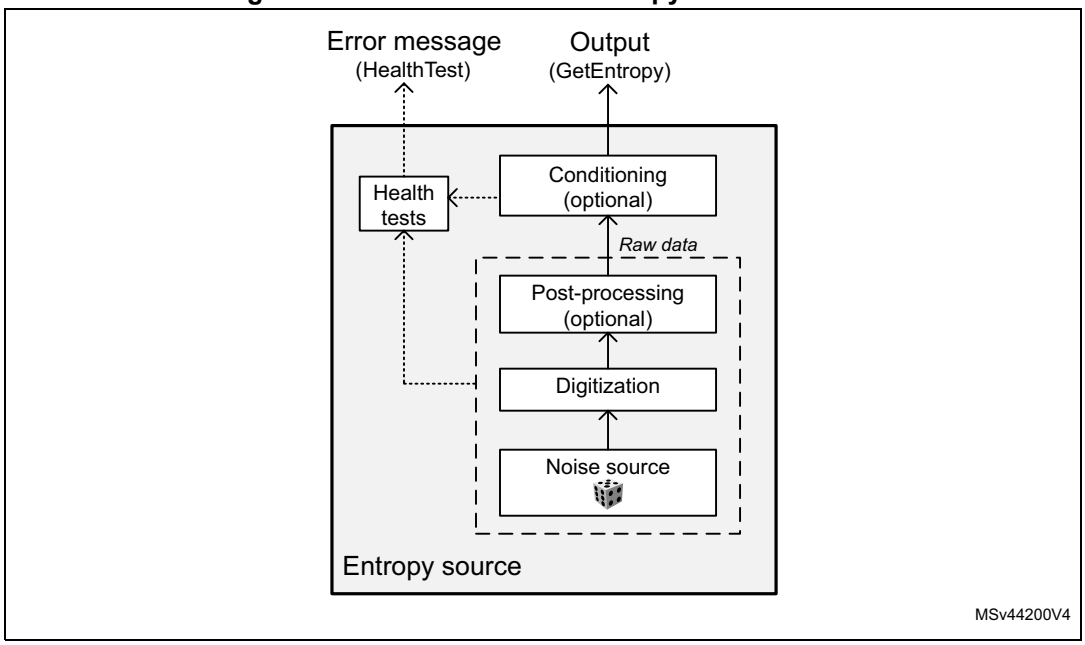
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk	Digital input	AHB clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk
rng_itamp_out	Digital output	RNG internal tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, RNG stops delivering random samples. A reset and a new initialization are needed to use it again.

31.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals.

Within its boundary, it integrates all the required NIST components shown in *Figure 417*: an analog noise source, a digitization stage, a conditioning algorithm, a health monitoring block, and two interfaces used to interact with the entropy source (GetEntropy and HealthTest).

Figure 417. NIST SP800-90B entropy source model



Noise source

This component contains the nondeterministic, entropy-providing activity, ultimately responsible for the uncertainty associated with the bit string output by the entropy source. This noise source provides 1-bit samples. It is composed of:

- Multiple analog noise sources (x3), each based on three XOR-ed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in *Section 31.3.8*. Multiple oscillators are also disabled for configuration A (see *Table 315*).
- The XOR-ing of all the noise sources into a single analog output.
- A sampling stage of this output is clocked by a dedicated clock input (rng_clk with integrated divider), delivering a 1-bit raw data output.

This noise source sampling is independent from the AHB interface clock frequency (rng_hclk), with the possibility, for the software, to decrease the sampling frequency by using the integrated divider.

Note: The recommended clock frequencies and associated divider value are detailed in Section 31.6.

Post processing

In the NIST configuration no post-processing is applied to the sampled noise source. In the non-NIST configuration B (as defined in *Section 31.6.2*) a normalization debiasing is applied: half of the bits are taken from the sampled noise source, the other half from the inverted sampled noise source.

Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bit string output (128-bit). The NIST SP800-90B target is full entropy on the output (128-bit).

The time intervals required between two random number generations, and between the RNG initialization and availability of the first sample are detailed in *Section 31.5*.

Output buffer

A data output buffer can store up to four 32-bit words that have been output from the conditioning component. When four words have been read from the output FIFO through the RNG_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register after a number of clock cycles specified in *Section 31.5*.

Whenever a random number is available through the RNG_DR register, the DRDY flag changes from 0 to 1. This flag remains high until the output buffer becomes empty after reading four words from the RNG_DR register.

Note: When the interrupts are enabled, an interrupt is generated when this flag transitions from 0 to 1. The interrupt is then cleared automatically by the RNG, as explained above.

Health checks

This component ensures that the entire entropy source (with its noise source) starts, then operates as expected, obtaining the assurance that failures are caught quickly, with high probability and reliability.

The RNG implements the following health check features in accordance with NIST SP800-90B. The described thresholds correspond to the default RNG_HTCR0 configuration.

1. Startup health tests, performed after reset and before the first use of the RNG as entropy source:
   - Repetition count test, flagging an error when the noise source has provided more than 28 consecutive bits at a constant value (0 or 1).
   - Adaptive proportion test running on a window of 1024 consecutive bits: the RNG verifies that the first bit on the outputs of the noise source is not repeated more than 684 times.
   - Known-answer tests, to verify the conditioning stage.
2. Continuous health tests, running indefinitely on the outputs of the noise source (XORed output and individual oscillator outputs, see *Figure 416: RNG block diagram*):
   - Repetition count test, similar to the one running in startup tests.
   - Adaptive proportion test, similar to the one running in startup tests.
3. Vendor specific continuous test:

- Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle (before divider) is smaller than the AHB clock cycle divided by 32.

4. On-demand test of digitized noise source (raw data)
   - Supported by restarting the entropy source and rerunning the startup tests (see software reset sequence in *Section 31.3.4*). Other kinds of on-demand testing (software based) are *not supported*.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in *Section 31.3.7*.

Note: An interrupt can be generated when an error is detected.

The health test thresholds are modified by changing the value in the corresponding RNG_HTCRx registers. See Section 31.6 for details.

31.3.4 RNG initialization

The RNG simplified state machine is shown in *Figure 418*.

After enabling the RNG (RNGEN = 1 in RNG_CR), the following chain of events occurs:

1. The analog noise source is enabled, and by default the RNG waits 16 RNG clock cycles (before divider) before starting to sample the analog output and filling the 128-bit conditioning shift register.
2. The conditioning hardware initializes, automatically triggering the startup behavior test on the raw data samples and known-answer tests.
3. Start-up health tests are completed, with the conditioning stage ready to use.
4. The conditioning stage internal input data buffer is filled again with 128-bit and a number of conditioning rounds defined by the RNG configuration (NIST or non-NIST) is performed. The output buffer is then filled with the post processing result.
5. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in *Section 31.5*.

Figure 418. RNG initialization overview

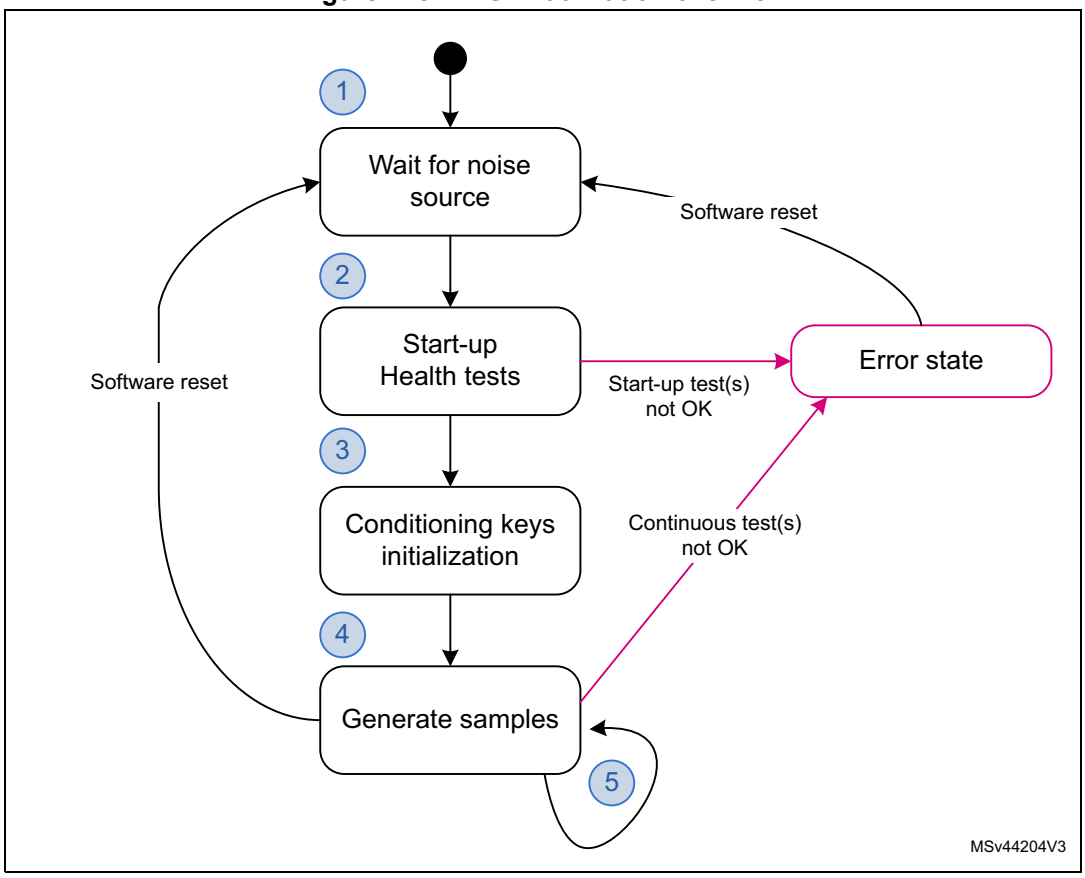


Figure 418 also highlights a possible software reset sequence, implemented by:

1. Write RNGEN = 0, then wait for BUSY = 0 in the RNG_SR register.
2. Write bits RNGEN = 0 and CONDRST = 1 in the RNG_CR register with the same RNG configuration and a new CLKDIV if needed.
3. Write RNGEN = 1 and CONDRST = 0 in the RNG_CR register.
4. Wait for a random number to be ready, after initialization completes.

Note: When the RNG is reset by RCC (hardware reset), the configuration for optimal randomness is lost in the registers.

31.3.5 RNG operation

Normal operation

To run the RNG using interrupts, the following steps are recommended:

1. Check in *Section 31.6* if a specific RNG configuration is required for the application.
   – If this is the case, when bits RNGEN and BUSY=0, write in the RNG_CR register bit CONDRST = 1, together with the correct RNG configuration. Then perform a second write with bit CONDRST = 0, interrupt enable bit IE = 1, and RNG enable bit RNGEN = 1.

- If this is not the case, perform a write to the RNG_CR register with interrupt enable bit IE = 1 and RNG enable bit RNGEN = 1.

2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore, at each interrupt, check that:
   - No error occurred. The SEIS and CEIS bits must be set to 0 in the RNG_SR register.
   - A random number is ready. The DRDY bit must be set to 1 in the RNG_SR register.
   - If the above two conditions are true the content of the RNG_DR register can be read up to four consecutive times. If valid data are available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in *Section 31.3.7* must be used.

To run the RNG in polling mode the following steps are recommended:

1. Check in *Section 31.6* if a specific RNG configuration is required for the application.
   - If this is the case, when bits RNGEN and BUSY = 0, write in the RNG_CR register bit CONDRST = 1, together with the correction RNG configuration. Then perform a second write to the RNG_CR register with bit CONDRST = 0 and the RNG enable bit RNGEN = 1.
   - If this is not the case, only enable the RNG by setting the RNGEN bit to 1 in the RNG_CR register.
2. Read the RNG_SR register and check that:
   - No error occurred (the SEIS and CEIS bits must be set to 0)
   - A random number is ready (the DRDY bit must be set to 1)
3. If the above conditions are true, read the content of the RNG_DR register up to four consecutive times. If valid data are available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, use the recovery sequence described in *Section 31.3.7*.

Note: When data are not ready (DRDY = 0) RNG_DR returns 0. It is recommended to verify that RNG_DR is different from 0. When this happens, a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

If the random number generation period is a concern to the application and if NIST compliance is not required, it is possible to select a faster configuration by using the configuration B, described in *Section 31.6*. The gain in random number generation speed is summarized in *Section 31.5*.

Low-power operation

If power consumption is a concern, low-power strategies can be used, as described in *Section 31.3.8*.

Software post-processing

No specific software post-processing/conditioning is expected to meet the AIS-31 or NIST SP800-90B approvals.

Built-in health check functions are described in *Section 31.3.3*.

31.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock.

The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock, coupled with a programmable divider (see CLKDIV bitfield in the RNG_CR register) is used for noise source sampling. Recommended clock configurations are detailed in *Section 31.6*.

*Note: When the CED bit in the RNG_CR register is set to 0, the RNG clock frequency before the internal divider **must be higher** than the AHB clock frequency divided by 32, otherwise the clock checker flags a clock error (CECS = 1 in the RNG_SR register).*

See *Section 31.3.1* for details (AHB and RNG clock domains).

31.3.7 Error management

In parallel to random number generation a health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

Clock error detection

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to 1 both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application must check that the RNG clock is configured correctly (see *Section 31.3.6*), and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when the clocking condition is normal.

When CECS is set, values are available in the RNG_DR register, but it must not be used as described in *Section 31.3.5*.

Note: CEIS is set only when CECS is set to 1 by RNG.

Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to 1 both SEIS and SECS bits to indicate that a seed error occurred.

When SECS is set, values are available in the RNG_DR register, but it must not be used as described in *Section 31.3.5*.

To manage a seed error, follow the recommended sequence below.

It requires the CONFIGLOCK bit to be cleared.

- When ARDIS is cleared (auto-reset enabled):
  - Clear the SEIS bit in the RNG_SR register
  - If SECS is cleared RNG can be used as usual. If SECS is still set in RNG_SR it means a new seed error occurred (unlikely). In this case, executes following step 1)
- When ARDIS is set (auto-reset disabled) the following sequence must be used to recover from a seed error.

1. Read both RNG_HTSRx registers. Using the previous HTSR values saved in RAM, compute the accumulated number of errors per oscillator (can be RPERR or ADERR errors). In a *nsmr* variable, mask the oscillators with accumulated number of errors greater than 3. This variable is used in step 3). Save the updated HTSR values in RAM.
   - During the first iteration, initialize *nsmr* variable with the RNG_NSCR value.
2. If fewer than 6 oscillators are active in the *nsmr* variable, reinitialize it with the RNG_NSCR value.
   - When RNG configuration is Configuration B, and the third time you reach this loop, go to Step 3 changing the RNG configuration to Configuration C. Both configurations are defined in *Section 31.6.2*.
3. Write CONDRST at 1 in RNG_CR, then update RNG_NSMR with *nsmr* value. Refer to *Section 31.7.8* for details.
4. Write CONDRST at 0 to finalize the reset of the RNG conditioning logic.
5. Wait for the CONDRST bit to be cleared by hardware, or poll for BUSY bit= 0. Then clear the SEIS interrupt status bit in the RNG_SR register.
6. If SECS is still set in RNG_SR a new seed error occurred (unlikely). In this case go back to step 1). If SECS is cleared, restart the generation of random numbers by setting RNGEN in RNG_CR.

If application has set the CONFIGLOCK bit (not recommended), do the following:

- When ARDIS is cleared (auto-reset enabled) clear the SEIS interrupt status bit in the RNG_SR register before drawing random numbers again.
- When ARDIS is set (auto-reset disabled), the following sequence must be used to fully recover from a seed error:

1. Write CONDRST at 1 and at 0 in the RNG_CR register to reset the conditioning logic.
2. Wait for the CONDRST bit to be cleared by hardware, or poll for BUSY = 0. Then clear the SEIS interrupt status bit in the RNG_SR register.
3. If SECS is still set in RNG_SR it means a new seed error occurred (unlikely). In this case go back to step 1).

Note: After a seed error RNG restarts generating random numbers when SECS is cleared.

RNG tamper errors

When an unexpected error is found by the RNG, an internal tamper event is triggered in the TAMP peripheral, and the RNG stops delivering random data.

When this event occurs, the secure application needs to reset the RNG peripheral, using the central reset management or the global SoC reset. A proper initialization of the RNG is required again.

Note: Seed errors do not trigger tamper errors.

31.3.8 RNG low-power use

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to 1 by setting the RNGEN bit to 0 in the RNG_CR register. As the post-processing logic and the output buffer remain operational while the RNGEN = 0, the following features are available to the software:

- If there are valid words in the output buffer, four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register, four additional random numbers can be still be read from the RNG_DR register. If it is not the case, RNG must be re-enabled by the application until the expected new noise source bits threshold is reached (128-bit in NIST mode), and a complete conditioning round is done. Four new random words are then available only if the expected number of conditioning round is reached (two if NISTC = 0). The overall time can be found in *Section 31.5*.

When disabling the RNG, the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (that is well before the DRDY bit rises for the first time), the initialization sequence resumes from where it was stopped when the RNGEN bit is set to 1, unless the application resets the conditioning logic using the CONDRST bit in the RNG_CR register.

When the application wants to gate the RNG kernel clock, the software must issue the following sequence: first, disable the RNG by clearing the RNGEN bit in the RNG_CR register, then wait for the BUSY bit to be cleared in the RNG_SR, and finally, disable the RNG kernel clock using the RCC.

Before entering a power mode where RNG is deactivated (like Stop mode), the software must first disable the RNG by clearing the RNGEN bit in the RNG_CR register, then wait for the BUSY bit to be cleared in the RNG_SR.

Note: The power modes where RNG is deactivated (retained or not available) can be found in the PWR section.

31.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see *Section 31.3.7*
- Clock error, see *Section 31.3.7*

Dedicated interrupt enable control bits are available as shown in *Table 314*.

Table 314. RNG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
RNG	Data ready flag	DRDY	IE	None (automatic)
	Seed error flag	SEIS	IE	Write CONDRST to 1 then to 0 unless ARDIS is cleared (see *Section 31.3.7*). Write 0 to SEIS.
	Clock error flag	CEIS	IE	Write 0 to CEIS

The user can enable or disable the above interrupt sources globally by using the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

Note: Interrupts are generated only when RNG is enabled.

31.5 RNG processing time

In the recommended configuration described in *Table 315*, the time between two sets of four 32-bit data is either:

- 206 x N AHB cycles if $f_{AHB} < f_{threshold}$ (conditioning stage is limiting), or
- 128 x N RNG cycles $f_{AHB} \geq f_{threshold}$ (noise source stage is limiting).

If f_{RNG}= 48 MHz, $f_{threshold}$ = 1.6 x f_{RNG} = 77 MHz. The value of N is defined in *Section 31.6*.

Note: When CLKDIV is different from 0, f_{RNG} must take into account the internal divider ratio.

31.6 RNG entropy source validation

31.6.1 Introduction

To assess the amount of entropy available from the RNG, the peripheral has been tested using the German BSI AIS-31 statistical tests (T0 to T8), and NIST SP800-90B test suite.

31.6.2 Validation conditions

The RNG true random number generator has been tested in the following conditions:

- RNG clock rng_clk = 48 MHz, with CED bit cleared in RNG_CR.
- Configurations are described in *Table 315* and *Table 316*. Only configuration A can be certified NIST SP800-90B. Refer to *Table 317* to select the best configuration for the application.
- Configuration B or C can be used when configuration A is not present in AN4230, or is not flagged as certified.

Table 315. RNG configurations

Configuration	RNG_CR bits					Loop number (N)	RNG_HTCR0 register	RNG_NSCR register
	NISTC bit	RNG_CONFIG1 [7:0]	CLKDIV [3:0]	RNG_CONFIG2 [2:0]	RNG_CONFIG3 [3:0](1)			
A	Refer to *NIST compliant RNG configuration* table in AN4230 available from *www.st.com*. This application note also indicates if this configuration is part of an existing NIST SP800-90B Entropy Certificate listed on *https://csrc.nist.gov/projects/cryptographic-module-validation-program.*							
B	1	0x83	0x0	0x0	0xF	3	0x0000 AAC7(2)	Default
C	1	0x84	0x5(3)	0x0	0xF	4		Default

1. RNG_CONFIG1[3:0] = i defines the loop number N. If i = 0, N =1. If NISTC = 0, N = 2 regardless of i.
2. This value can be fixed in the RNG driver (it does not depend upon the STM32 product).
3. The noise source sampling must be 48 MHz or less. If the RNG clock is different from 48 MHz, this value of CLKDIV must be adapted. See the CLKDIV bitfield description in *Section 31.7.1* for details.

Table 316. Additional health test configurations

Configuration	RNG_HTCR1 register (oscillator set 1)	RNG_HTCR2 register (oscillator set 2)	RNG_HTCR3 register (oscillator set 3)
A	Refer to NIST compliant RNG configuration table in AN4230 available from *www.st.com*.		
B and C	Default	Default	Default

Table 317. Configuration selection

Section criteria	Configuration A	Configuration B	Configuration C
Suitable to generate NIST compliant cryptographic keys	Yes	No	
Entropy(1)	Certified	Not certified	
Speed(2)	Baseline	Faster	Baseline

1. For configurations B and C entropy is verified using the AIS-31 test suite (T0 to T8).
2. When the speed is not enough for the application, an NIST compliant DRBG can be used to increase throughput.

For details on data collection and the running of statistical test suites refer to AN4230 "*Introduction to random number generation validation using the NIST statistical test suite for STM32 MCUs and MPUs*", available on *www.st.com*.

31.7 RNG registers

The RNG is associated with a control register, a data register, and a status register.

31.7.1 RNG control register (RNG_CR)

Address offset: 0x000

Reset value: 0x0080 0D00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CONFIGLOCK	CONDRST	Res.	Res.	RNG_CONFIG1[7:0]								CLKDIV[3:0]			
rs	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNG_CONFIG2[2:0]			NISTC	RNG_CONFIG3[3:0]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw		

Bit 31 **CONFIGLOCK**: RNG configuration lock

Set this bit to lock the RNG configuration. Since CONFIGLOCK prevents the implementation of robust seed error management described in *Section 31.3.7*, it is recommended not to use it.

0: Writes to the RNG_NSMR, RNG_NSCR, RNG_HTCR and the RNG_CR configuration bits [29:4] are allowed.

1: Writes to the RNG_NSMR, RNG_NSCR, RNG_HTCR and the RNG_CR configuration bits [29:4] are ignored until the next RNG reset.

Once set, this bit can only be cleared when RNG is reset (set once bit).

Bit 30 **CONDRST**: Conditioning soft reset

Write 1 and then write 0 to reset the conditioning logic, clear all the FIFOs and start a new RNG initialization process, with RNG_SR cleared. Registers RNG_NSMR, RNG_NSCR, RNG_CR and RNG_HTCR are not changed by CONDRST.

This bit must be set to 1 in the same access that sets any configuration bits [29:4]. In other words, when CONDRST bit is set to 1 correct configuration in bits [29:4] must also be written.

Before setting this bit to 1 RNGEN must be set to 0, and BUSY flag must be cleared in RNG_SR.

When CONDRST is set to 0, wait for BUSY flag to be cleared in RNG_SR. CONDRST bit then reads as 0.

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:20 **RNG_CONFIG1[7:0]:** RNG configuration 1

Reserved to the RNG configuration (bitfield 1). Must be initialized using the recommended value documented in *Section 31.6*.

Writing to this bitfield is only effective when CONFIGLOCK = 0. The new value is only taken into account if CONDRST bit is set in the same access.

Bits 19:16 **CLKDIV[3:0]**: Clock divider factor

This value used to configure an internal programmable divider (from 1 to 16) acting on the incoming RNG clock. These bits can be written only when the core is disabled (RNGEN = 0).

0x0: internal RNG clock after divider is similar to incoming RNG clock.

0x1: two RNG clock cycles per internal RNG clock.

0x2: 2^2 (= 4) RNG clock cycles per internal RNG clock.

...

0xF: 2^{15} RNG clock cycles per internal clock (for example. an incoming 48 MHz RNG clock becomes a 1.5 kHz internal RNG clock)

Writing to this bitfield is only effective when CONFIGLOCK = 0. The new value is only taken into account if CONDRST bit is set in the same access.

Bits 15:13 **RNG_CONFIG2[2:0]:** RNG configuration 2

Reserved to the RNG configuration (bitfield 2). Refer to the RNG_CONFIG1 bitfield for details.

Bit 12 **NISTC**: NIST custom

0: Hardware default values for NIST compliant RNG. In this configuration for 128-bit output, two conditioning loops are performed and 256 bits of noise source are used.

1: Custom values for NIST compliant RNG. See *Section 31.6* for recommended configurations.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bits 11:8 **RNG_CONFIG3[3:0]:** RNG configuration 3

Reserved to the RNG configuration (bitfield 3). Refer to RNG_CONFIG1 bitfield for details.

If the NISTC bit is cleared in this register RNG_CONFIG3 bitfield values are ignored by RNG.

Bit 7 **ARDIS**: Auto reset disable

Set this bit to deactivate the auto-reset feature.

0: Auto-reset is enabled

1: Auto-reset is disabled

Keeping the auto-reset enabled (automatic clearance of the SECS bit) simplifies the management of noise source errors, as described in *Section 31.3.7*.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CED:** Clock error detection

0: Clock error detection is enabled.

1: Clock error detection is disabled

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, that is to enable or disable CED, the RNG must be disabled.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE:** Interrupt enable

0: RNG interrupt is disabled

1: RNG interrupt is enabled. An interrupt is pending as soon as the DRDY, SEIS, or CEIS flag is set in the RNG_SR register.

Bit 2 **RNGEN:** True random number generator enable
This bit enables/disables the true random number generator.
0: True random number generator disabled.
1: True random number generator enabled.
Clearing RNGEN powers off the analog noise sources and gates the logic clocked by the RNG clock.
When clearing this bit verify that BUSY flag is cleared in RNG_SR before updating this register again.

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

31.7.2 RNG status register (RNG_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	BUSY	Res.	SECS	CECS	DRDY
									rc_w0	rc_w0	r		r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SEIS:** Seed error interrupt status
This bit is set at the same time as SECS. It is cleared by writing 0 (unless CONDRST is used). Writing 1 has no effect.
0: No faulty sequence detected
1: At least one faulty sequence is detected. See SECS bit description for details.
An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 5 **CEIS:** Clock error interrupt status
This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.
0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$)
1: The RNG clock before the internal divider is detected too slow ($f_{RNGCLK} < f_{HCLK}/32$)
An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 4 **BUSY:** Busy
This flag indicates whether RNG is idle or busy.
0: Idle
1: Busy
RNG is flagged as busy when a conditioning soft reset (using CONDRST) is in progress. It is also flagged busy when RNG is activated by application setting RNGEN bit, or when a connected peripheral fetches randoms via the dedicated RNG hardware bus.
More information on BUSY usage can also be found in *Section 31.3.8*.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SECS:** Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: At least one of the following faulty sequence has been detected:

– Start-up or continuous repetition count test on noise source failed.

– Start-up or continuous adaptive proportion test on noise source failed.

– Start-up post-processing/conditioning sanity check failed.

After enabling the RNG, SECS bit clears once startup health tests completes successfully. If SECS does not clear, run the software reset sequence with CONDRST described in *Section 31.3.4*.

Bit 1 **CECS:** Clock error current status

0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.

1: The RNG clock is too slow ($f_{RNGCLK} < f_{HCLK}/32$).

CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.

Bit 0 **DRDY:** Data ready

This flag is raised to indicate valid (non-0) random data available in the RNG_DR register. It is cleared by hardware upon reading the RNG_DR register (which empties the RNG buffer). The DRDY flag can also be raised upon disabling RNG (upon clearing the RNGEN bit).

0: No valid random data ready.

1: Valid random data ready.

Raising DRDY generates an interrupt if enabled with the IE bit of the RNG_CR register.

31.7.3 RNG data register (RNG_DR)

Address offset: 0x008

Reset value: 0x0000 0000

This register is a read-only register that delivers a 32-bit random value when read. The content of this register is valid when the DRDY = 1 and the value is not 0x0, even if RNGEN = 0.

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

Bits 31:0 **RNDATA[31:0]:** Random data

32-bit random data, valid when DRDY = 1. When DRDY = 0, the RNDATA value is 0.

The 0 value means that a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

31.7.4 RNG noise source control register (RNG_NSCR)

Address offset: 0x00C

Reset value: 0x0000 01FF

Writing in RNG_NSCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in RNG_CR. Writing to this register is ignored if CONFIGLOCK= 1.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC3[2:0]			EN_OSC2[2:0]			EN_OSC1[2:0]		
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:6 **EN_OSC3[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 3. The bitfield has no effect otherwise.

Bits 5:3 **EN_OSC2[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 2. The bitfield has no effect otherwise.

Bits 2:0 **EN_OSC1[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 1. The bitfield has no effect otherwise.

31.7.5 RNG health test control register x (RNG_HTCRx)

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

Reset value: 0x0000 72AC, 0x0003 FFFF, 0x0003 FFFF, 0x0003 FFFF

Writing in RNG_HTCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in the RNG_CR. Writing to this register is ignored if CONFIGLOCK = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HTCFG[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 31:0 **HTCFG[31:0]:** health test configuration

This configuration is used by RNG to configure the health tests (repetition count test, adaptive proportion test) at position x. See *Section 31.6* for the recommended value. With its default value, health tests have a very small probability of triggering any error at position x.

Note: The RNG behavior, including the read to this register, is not guaranteed if a different value from the recommended value is written.

31.7.6 RNG health test status register 0 (RNG_HTSR0)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RPERR9	RPERR8	RPERR7	RPERR6	RPERR5	RPERR4	RPERR3	RPERR2	RPERR1	RPERRX
						r	r	r	r	r	r	r	r	r	r

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

Bits 9:1 **RPERR{i}:** Repetitive error for oscillator *i* (i=1 to 9)

This bit is set when the health test logic associated to the oscillator *i* finds a repetitive test error. The bit is cleared when application clears SEIS bit in RNG_SR. If ARDIS is cleared auto-reset prevents RNG to capture more than one test error. Repetition count test is defined in *Health checks* section.

0: No repetitive test error found in oscillator *i*.
1: Repetitive test error found in oscillator *i*.

Bit 0 **RPERRX:** Repetitive error after the XOR

This bit is set when the health test logic finds a repetitive test error after the XOR clock tree (see *Figure 416*). The bit is cleared when application clears SEIS bit in RNG_SR. If ARDIS is cleared auto-reset prevents RNG to capture more than one test error.

0: No repetitive test error found after the XOR clock tree.
1: Repetitive test error found after the XOR clock tree.

31.7.7 RNG health test status register 1 (RNG_HTSR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	ADERR9	ADERR8	ADERR7	ADERR6	ADERR5	ADERR4	ADERR3	ADERR2	ADERR1	ADERRX
						r	r	r	r	r	r	r	r	r	r

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

Bits 9:1 **ADERR{i}:** Adaptative error for oscillator *i* (i=1 to 9)

This bit is set when the health test logic associated to the oscillator *i* finds an adaptative test error. The bit is cleared when application clears SEIS bit in RNG_SR. If ARDIS is cleared auto-reset prevents RNG to capture more than one test error. Adaptive proportion test is defined in *Health checks* section.

0: No adaptative test error found in oscillator *i*.
1: Adaptative test error found in oscillator *i*.

Bit 0 **ADERRX:** Adaptative error after the XOR

This bit is set when the health test logic finds an adaptative test error after the XOR clock tree (see *Figure 416*). The bit is cleared when application clears SEIS bit in RNG_SR If ARDIS is cleared auto-reset prevents RNG to capture more than one test error.

0: No adaptative test error found after the XOR clock tree.
1: Adaptative test error found after the XOR clock tree.

31.7.8 RNG noise source mask register (RNG_NSMR)

Address offset: 0x030

Reset value: 0x0000 01FF

Writing to this register is ignored if CONFIGLOCK = 1. It is recommended to have at least 6 oscillators running when loop number N = 3 (or higher). See *Section 31.6.2* for details.

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

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **MOSC{i}:** Mask oscillator i (i = 1 to 9)

Clear this bit to masks the oscillator *i*, and deactivate its related health tests. The bit has no effect if the oscillator *i* is deactivated with the bit RNG_NSCR[i].

Application can mask an oscillator when it is not contributing to the generation of entropy.

0: Oscillator *i* is masked, and related health tests are deactivated.
1: Oscillator *i* is not masked. Related health tests are running.

31.7.9 RNG register map

Table 318. RNG register map and reset map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RNG_CR	CONFIGLOCK	CONDRST	Res.	Res.	RNG_CONFIG1[7:0]								CLKDIV[3:0]				RNG_ CONFIG2 [2:0]			NISTC	RNG_ CONFIG3 [3:0]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
	Reset value	0	0			0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0		0		0	0		
0x004	RNG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	BUSY	Res.	SECS	CECS	DRDY
	Reset value																										0	0	0		0	0	0
0x008	RNG_DR	RNDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	RNG_NSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC3 [2:0]			EN_OSC2 [2:0]			EN_OSC1 [2:0]		
	Reset value																								1	1	1	1	1	1	1	1	1
0x010	RNG_HTCR0	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	1	0	1	1	0	0
0x014	RNG_HTCR1	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x018	RNG_HTCR2	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x01F	RNG_HTCR3	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x020	RNG_HTSR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPERR9	RPERR8	RPERR7	RPERR6	RPERR5	RPERR4	RPER3	RPERR2	RPERR1	RPERRX
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x024	RNG_HTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADERR9	ADERR8	ADERR7	ADERR6	ADERR5	ADERR4	ADERR3	ADERR2	ADERR1	ADERRX
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x030	RNG_NSMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MOSC9	MOSC8	MOSC7	MOSC6	MOSC5	MOSC4	MOSC3	MOSC2	MOSC1
	Reset value																								1	1	1	1	1	1	1	1	1

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

32 Hash processor (HASH)

32.1 HASH introduction

The hash processor is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA-2 family) and the HMAC (keyed-hash message authentication code) algorithm. HMAC is suitable for applications requiring message authentication.

The hash processor computes FIPS (Federal Information Processing Standards) approved digests of length of 160, 224, 256 bits, for messages of any length less than 2^{64} bits (for SHA-1, SHA-224 and SHA-256) or less than 2^{128} bits (for SHA-384, SHA-512).

32.2 HASH main features

- Suitable for data authentication applications, compliant with:
  - Federal Information Processing Standards Publication FIPS PUB 180-4, *Secure Hash Standard* (SHA-1 and SHA-2 family)
  - Federal Information Processing Standards Publication FIPS PUB 186-4, *Digital Signature Standard (DSS)*
  - Internet Engineering Task Force (IETF) Request For Comments RFC 2104, *HMAC: Keyed-Hashing for Message Authentication* and Federal Information Processing Standards Publication FIPS PUB 198-1, *The Keyed-Hash Message Authentication Code (HMAC)*
- Fast computation of SHA-1, SHA2-224, SHA2-256, SHA2-384, and SHA2-512
  - 82 (respectively 66) clock cycles for processing one 512-bit block of data using SHA-1 (respectively SHA-256) algorithm
  - 98 clock cycles for processing one 1024-bit block of data using either SHA2-384 or SHA2-512 algorithm
  - Support for SHA-2 truncated outputs (SHA2-512/224, SHA2-512/256)
- Support for HMAC mode with all supported algorithm
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
  - Automatic 32-bit words swapping to comply with the internal little-endian representation of the input bit-string
  - Supported word swapping format: bits, bytes, half-words, and 32-bit words
- Single 32-bit, write-only, input register associated to an internal input FIFO that can receive a single data block of size depending on the selected algorithm
- Automatic padding to complete the input bit string to fit digest minimum block size
- AMBA AHB target peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
- 8 × 32-bit words (H0 to H15) for output message digest
- Automatic data flow control supporting direct memory access (DMA) using one channel.
- Support for both single and fixed DMA burst transfers of four words.
- Interruptible message digest computation, on a per-block basis
  - Reloadable digest registers

– Hashing computation suspend/resume mechanism, including DMA

32.3 HASH implementation

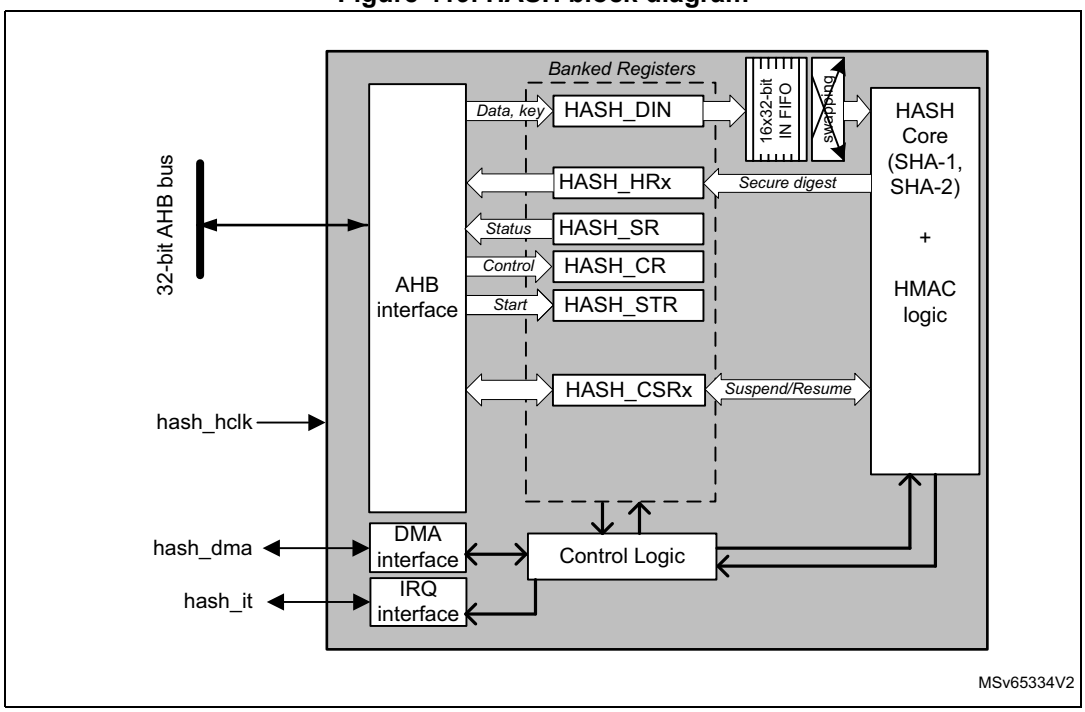
The devices have a single instance of HASH peripheral.

32.4 HASH functional description

32.4.1 HASH block diagram

Figure 419 shows the block diagram of the hash processor.

Figure 419. HASH block diagram



32.4.2 HASH internal signals

Table 319 describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

Table 319. HASH internal input/output signals

Signal name	Signal type	Description
hash_hclk	digital input	AHB bus clock
hash_it	digital output	Hash processor global interrupt request
hash_dma	digital input/output	DMA burst request/ acknowledge

32.4.3 About secure hash algorithms

The hash processor is a fully compliant implementation of the secure hash algorithm defined by FIPS PUB 180-4 standard.

With each algorithm, the HASH computes a condensed representation of a message or data file. More specifically, when a message is presented on the input, the HASH processing core produces a fixed-length output string called a message digest (see *Table 320*).

Table 320. Information on supported hash algorithms

Algorithm	Message digest size (in bits)	Block size (in bytes)(1)	Message length	Bit string message
SHA-1	160	64	$< 2^{64}$ bits	Yes
SHA2-224	224			
SHA2-256	256			
SHA2-384	384	128	$< 2^{128}$ bits	Yes
SHA2-512	512(2)			

1. Block size = (NBWE-1) * 4 bytes. NBWE[4:0] bitfield can be read from the HASH_SR register after ALGO and INIT bits are written in the HASH_CR register.
2. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes).

The message digest can then be processed with a digital signature algorithm in order to generate or verify the signature for the message.

Signing the message digest rather than the message often improves the efficiency of the process since the message digest is usually much smaller in size than the message. The verifier of a digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-2 functions supported by the hash processor are qualified as “secure” by NIST because it is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest (SHA-1 does not qualify as secure since February 2017).

32.4.4 Message data feeding

The message (or data file) to be processed by the HASH must be considered as a bit string. Per FIPS PUB 180-4 standard this message bit string grows from left to right, with hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most bit position. For example message string “abc” with a bit string representation of “**01100001 01100010 01100011**” is represented by a 32-bit word **0x00636261**, and 8-bit words **0x61626300**.

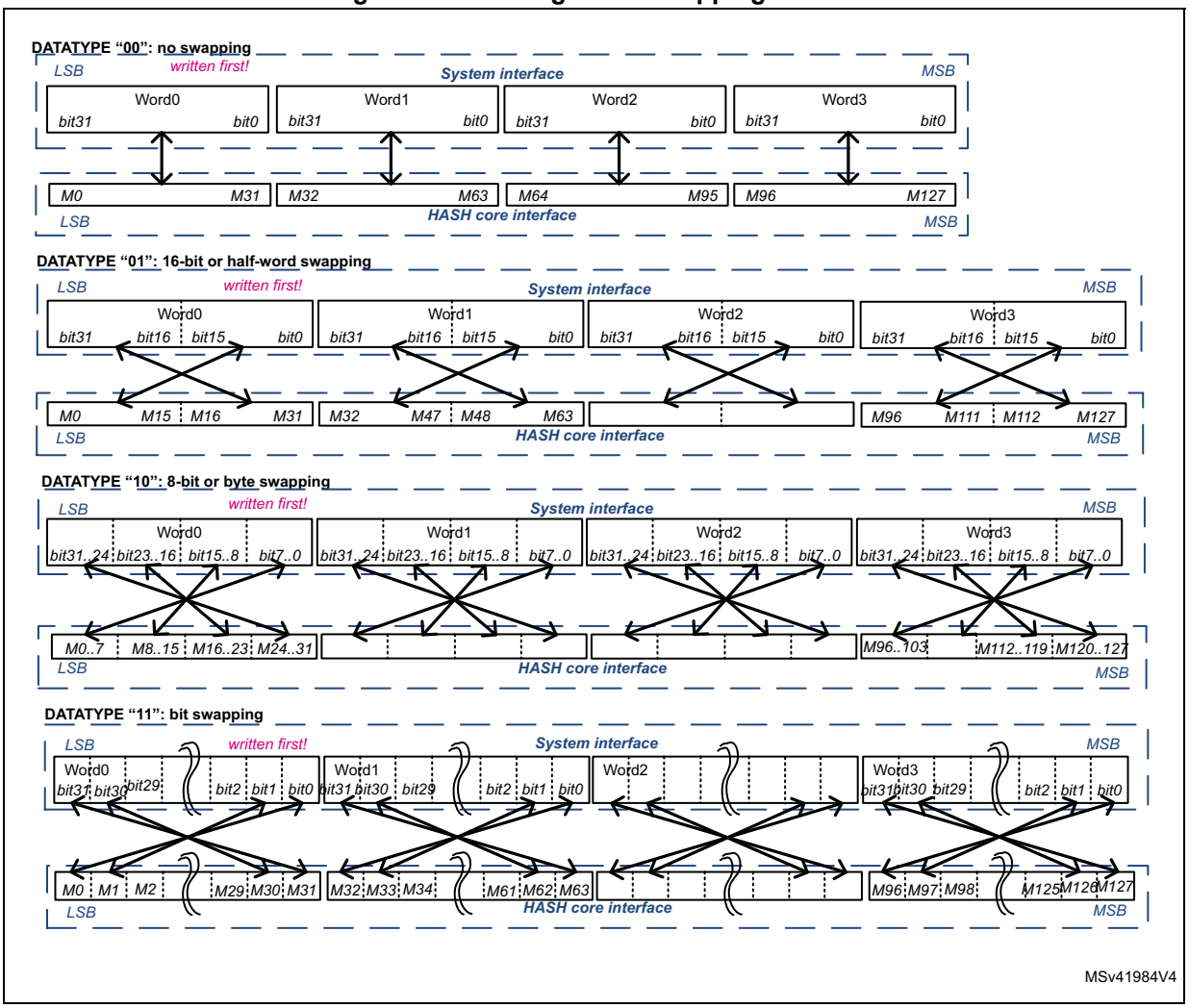
Data are entered into the HASH one 32-bit word at a time, by writing them into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the 16 words input FIFO each time the register is written with new data. Hence, HASH_DIN and the FIFO form a seventeen 32-bit words length FIFO (named the IN buffer).

In accordance to the kind of data to be processed (for example byte swapping when data are ASCII text stream) there must be a bit, byte, half-word, or no swapping operation to be performed on data from the input FIFO before entering the little-endian hash processing core. *Figure 420* shows how the hash processing core 32-bit data block M0...31 is

constructed from one 32-bit words popped into input FIFO by the driver, according to the DATATYPE bitfield in the HASH control register (HASH_CR)

HASH_DIN data endianness when bit swapping is disabled (DATATYPE = 00) can be described as following: the least significant bit of the message has to be at MSB position in the first word entered into the hash processor, the 32nd bit of the bit string has to be at MSB position in the second word entered into the hash processor and so on.

Figure 420. Message data swapping feature



32.4.5 Message digest computing

The hash processor sequentially processes several blocks when computing the message digest. Block sizes can be found in *Table 320: Information on supported hash algorithms*.

Each time the DMA or the CPU writes a block to the hash processor, the HASH automatically starts computing the message digest. This operation is known as partial digest computation.

As described in *Section 32.4.4: Message data feeding*, the message to be processed is entered into the HASH 32-bit word at a time, writing to the HASH_DIN register to fill the

input FIFO. In order to perform the hash computation on this data the application must follow below sequence.

1. Initialize the hash processor using the HASH_CR register:
   - Write to the HASH_CR register to select the right algorithm using the ALGO bitfield, and set the INIT bit (other bits are kept at zero). Then read the NBWE bitfield from the HASH_SR register to deduce the algorithm block size, which equals **(NBWE-1) * 4**. This step is not required if the block size is already known (see *Table 320* for details).
   - Select the right algorithm using the ALGO[3:0] field. If needed, program the correct swapping operation on the message input words using the DATATYPE[1:0] bitfield.
   - When HMAC mode is required, set the MODE bit as well as the LKEY bit if the HMAC key size is greater than the known block size of the algorithm (otherwise keep LKEY cleared). Refer to *Section 32.4.7: HMAC operation* for details.
   - Update NBLW[4:0] in the HASH_STR register to define the number of valid bits in the last word of the message if it is different from 32 bits. NBLW information is used to correctly perform the automatic message padding before the final message digest computation.
2. Complete the initialization by setting the INIT bit in HASH_CR register. Also set the DMAE bit if data are transferred via DMA.

Caution: When programming step 2, it is important that the correct configuration values (ALGO, DATATYPE, HMAC mode, key length, NBLW) are set up before or at the same time.

3. Start filling data by writing to the HASH_DIN register, unless data are automatically transferred via DMA. Note that the processing of a block can start only once the last value of the block has entered the input FIFO. The way the partial or final digest computation is managed depends on the way data are fed into the processor:
   - Data are filled by software:

     Partial digest computations are triggered each time the application writes the first word of the next block, the block size being defined by NBWE bits in HASH_SR. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write new data to HASH_DIN. This mechanism avoids the introduction of wait states by the HASH.

     The final digest computation is triggered when the last block is entered and the software sets the DCAL bit. If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to writing DCAL bit (see *Section 32.4.6* for details).
   - Data are filled as a single DMA transfer (MDMAT = 0):

     Partial digest computations are triggered automatically each time the FIFO is full.

     The final digest computation is triggered automatically when the last block has been transferred to the HASH_DIN register by DMA (DCAL bit is set by hardware). If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to enabling the DMA (see *Section 32.4.6* for details).
   - Data are filled using multiple DMA transfers (MDMAT = 1):

     Partial digest computations are triggered as for single DMA transfers (refer to the above description). However, the final digest computation is not triggered automatically when the last block has been transferred by DMA to the HASH_DIN register (DCAL bit is not set by hardware). It enables the hash processor to

receive a new DMA transfer as part of this digest computation. To launch the final digest computation, the software must clear MDMAT bit before the last DMA transfer in order to trigger the final digest computation as it is done for single DMA transfers.

4. Once the digest calculation is completed (DCIS = 1), the resulting digest can be read from the output registers, as described in *Table 321*. When set, the DMAEN bit is automatically cleared by hardware.

Table 321. Hash processor outputs

Algorithm	Valid output registers(1)	Most significant bit	Digest size (in bits)
SHA-1	HASH_H0 to HASH_H4	HASH_H0[31]	160
SHA2-224	HASH_H0 to HASH_H6	HASH_H0[31]	224
SHA2-256	HASH_H0 to HASH_H7		256
SHA2-384	HASH_H0 to HASH_H11	HASH_H0[31]	384
SHA2-512	HASH_H0 to HASH_H15		512(2)

1. Aliased HASH_HRAx registers are not used with SHA-3 or SHAKE algorithm (registers read as zero).
2. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes)

For more information about HMAC detailed instructions, refer to *Section 32.4.7: HMAC operation*.

Aborting HASH operation

HASH can be disabled at any time by writing 0x4 to HASH_CR, after stopping any running DMA transfer to HASH (if applicable).

32.4.6 Message padding

Overview

When computing a condensed representation of a message, the process of feeding data into the hash processor (with automatic partial digest computation every block size transfer) loops until the last bits of the original message are written to the HASH_DIN register.

As the length (number of bits) of a message can be any integer value, the last word written to the hash processor may have a valid number of bits between 1 and 32. This number of valid bits in the last word, NBLW, has to be written to the HASH_STR register, so that message padding is correctly performed before the final message digest computation.

Padding processing

Detailed hashing sequences with padding are described in *Section 32.4.5: Message digest computing*. As mentioned in these sequences, the padding is done automatically using the information provided in NBLW[4:0].

Hashing example

Below is an example of SHA-1 digest computation with HASH. Message processing and padding are defined in Federal Information Processing Standards PUB 180-4.

Let us assume that the original message is the ASCII binary-coded form of “abc”, of length L = 24 (‘**U**’ means don’t care):

```
byte 0    byte 1   byte 2   byte 3
01100001 01100010 01100011 UUUUUUUU
<-- 1st word written to HASH_DIN -->
```

NBLW has to be loaded with the value 24: a “**1**” is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH_DIN register (little-endian convention):

```
01100001 01100010 01100011 1UUUUUUU
```

Since L = 24, the number of bits in the above bit string is 25, and 423 “0” bits are automatically appended, making now 448 bits.

This gives in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000
```

The message length value, **L**, in two-word format (that is 00000000 00000018) is appended. Hence, the final padded message in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018
```

If the hash processor is programmed to swap byte within HASH_DIN input register (DATATYPE = 10 in HASH_CR), the above message has to be entered using following sequence:

1. **0xUU636261** is written to the HASH_DIN register (where ‘**U**’ means don’t care)
2. **0x18** is written to the HASH_STR register (the number of valid bits in the last word written to the HASH_DIN register is 24, as the original message length is 24 bits)
3. **0x100** is written to the HASH_STR register to run the final digest computation.
4. The hash computing is complete with the message digest available in the HASH_HRx registers (x = 0 to 4) for the SHA-1 algorithm. For this FIPS example, the expected value is as follows:

```
HASH_HR0 = 0xA9993E36
HASH_HR1 = 0x4706816A
HASH_HR2 = 0xBA3E2571
HASH_HR3 = 0x7850C26C
HASH_HR4 = 0x9CD0D89D
```

32.4.7 HMAC operation

Overview

As specified by Internet Engineering Task Force RFC2104 and NIST FIPS PUB 198-1, the HMAC algorithm is used for message authentication by irreversibly binding the message being processed to a key chosen by the user. The algorithm consists of two nested hash operations:

```
HMAC(message)= Hash((Key | pad) XOR opad |
               Hash((Key | pad) XOR ipad | message))
```

Where:

- **opad** = $[0x5C]_n$ (outer pad) and **ipad** = $[0x36]_n$ (inner pad)
- $[X]_n$ represents a repetition of X *n* times, where *n* equal to the byte size of the underlying hash function data block (n = 64 when block size is 512 bits).
- **pad** is a sequence of zeroes needed to extend the key to the length *n* defined above. If the key length is greater than *n*, the application must first hash the key using Hash() function and then use the resultant byte string as the actual key to HMAC.
- | represents the concatenation operator.

Note: HMAC mode of the hash processor can be used with all supported algorithms.

HMAC processing

Four different steps are required to compute the HMAC:

1. The software sets the INIT bit, with the MODE bit set and the ALGO bits selecting the desired algorithm. The LKEY bit must also be set if the key being used is longer than the block size. In this case, as required by HMAC specifications, the hash processor uses the hash of the key instead of the real key.
2. The software provides the key to be used for the inner hash function, using the same mechanism as the message string loading that is by writing the key data into HASH_DIN register and then completing the transfer by setting DCAL bit and the correct NBLW to HASH_STR register.
3. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write the message string to HASH_DIN. When the last word of the last block is entered and the software sets DCAL bit in HASH_STR register, the NBLW bitfield must be programmed at the same time to a value different from zero if the message length is not an exact multiple of the block size. Note that the DMA can also be used to feed the message string, as described in *Section 32.4.5: Message digest computing*.
4. Once the processor is ready again (DINIS = 1 in HASH_SR), the software provides the key to be used for the outer hash function, writing the key data into HASH_DIN register, and then completing the transfer by setting DCAL bit and programming the correct NBLW to HASH_STR register. The HMAC result can be found in the valid output registers (HASH_HRx) as soon as DCIS bit is set.

Note: The computation latency of the HMAC primitive depends on the lengths of the keys and message, as described in Section 32.4.11: HASH processing time.

Endianness management details can be found in Section 32.4.4: Message data feeding.

HMAC example

Below is an example of HMAC SHA-1 algorithm (ALGO = 00 and MODE = 1 in HASH_CR) as specified by NIST. SHA-1 block size is 64 bytes.

Let us assume that the original message is the ASCII binary-coded form of "**`Sample message for keylen = blocklen`**", of length L = 34 bytes. If the HASH is programmed in no swapping mode (DATATYPE = 00 in HASH_CR), the following data must be loaded sequentially into HASH_DIN register:

1. **Inner hash key** input (length = 64, that is, no padding), specified by NIST. As key length = 64, LKEY bit is cleared in HASH_CR register

   ```
   00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617
   18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F
   30313233 34353637 38393A3B 3C3D3E3F
   ```

2. **Message** input (length = 34, that is, padding required). HASH_STR must be set to **`0x20`** to start message padding and inner hash computation (see **'U**' as don't care)

   ```
   53616D70 6C65206D 65737361 67652066 6F72206B 65796C65
   6E3D626C 6F636B6C 656EUUUU
   ```

3. **Outer hash key** input (length = 64, that is, no padding). A key identical to the inner hash key is entered here.
4. **Final outer hash computing** is then performed by the HASH. The HMAC-SHA1 digest result is available in the HASH_HRx registers, as shown below:

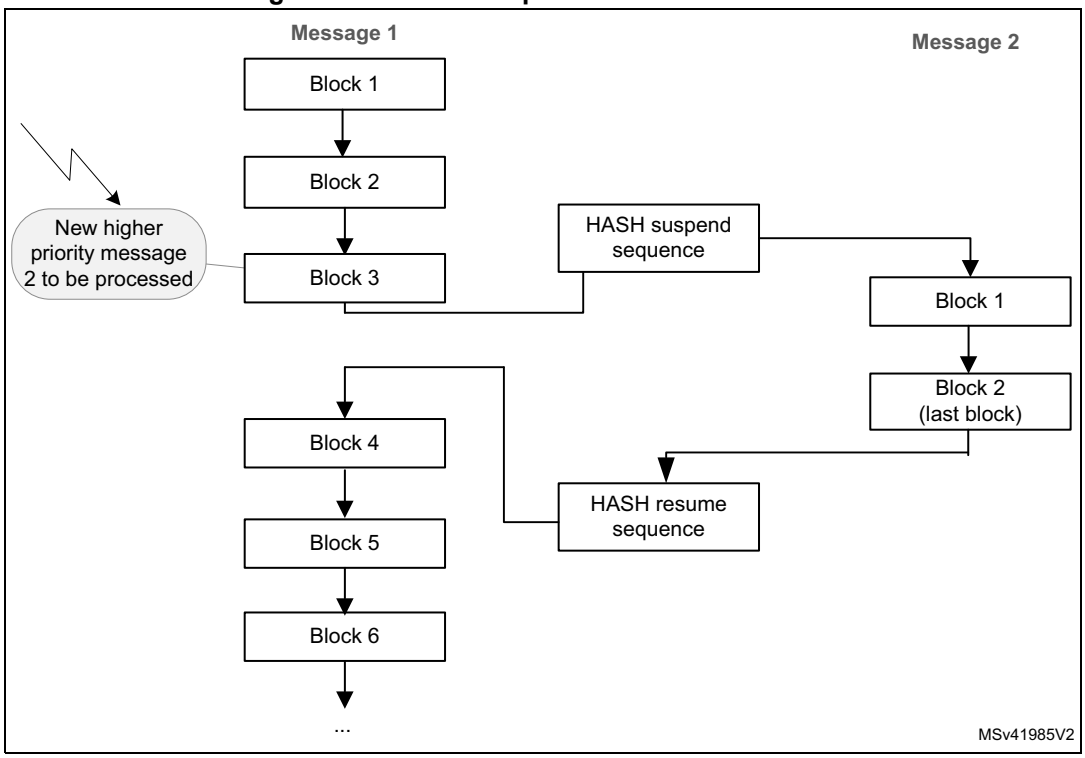
   ```
   HASH_HR0 = 0x5FD596EE
   HASH_HR1 = 0x78D5553C
   HASH_HR2 = 0x8FF4E72D
   HASH_HR3 = 0x266DFD19
   HASH_HR4 = 0x2366DA29
   ```

32.4.8 HASH suspend/resume operations

Overview

It is possible to interrupt a hash/HMAC operation to perform another processing with a higher priority. The interrupted process completes later when the higher-priority task has been processed, as shown in *Figure 421*.

Figure 421. HASH suspend/resume mechanism



To do so, the context of the interrupted task must be saved from the HASH registers to memory, and then be restored from memory to the HASH registers.

The procedures where the data flow is controlled by software or by DMA are described hereafter.

Data loaded by software

When the DMA is not used to load the message into the hash processor, the context can be saved only when no block processing is ongoing.

To suspend the processing of a message, proceed as follows after writing the number of words defined in NBWE:

1. In Polling mode, wait for BUSY = 0 then poll if the DINIS status bit is set.
   In Interrupt mode, implement the next step in DINIS interrupt handler (recommended).
2. Store the contents of the following registers into memory:
   - HASH_IMR
   - HASH_STR
   - HASH_CR
   - HASH_CSR0 to HASH_CSR37, when SHA-1 or SHA2-256 is selected, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing
   - HASH_CSR0 to HASH_CSR90, when SHA2-384 or SHA2-512 (truncated or not) is selected, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.

To resume the processing of a message, proceed as follows:

1. Write the following registers with the values saved in memory: HASH_IMR, HASH_STR, HASH_CR.
2. Initialize the hash processor by setting the INIT bit in the HASH_CR register
3. Write the HASH_CSRx registers with the values saved in memory.
4. Restart the processing from the point where it has been interrupted.

Data loaded by DMA

When the DMA is used to load the message into the hash processor, it is recommended to suspend and then restore a secure digest computing as described below.

In this sequence the DMA channel allocated to the hash peripheral remains allocated to the processing of message 1 (see *Figure 421*).

To suspend the processing of a message using DMA, proceed as follows:

1. Clear the DMAE bit to disable the DMA interface. The hash peripheral automatically fetches enough data via the DMA to complete the current burst transfer.
2. Wait until the last DMA transfer is complete (DMAS = 0 in HASH_SR).
3. Disable the DMA channel.
4. In Polling or Interrupt mode (recommended), wait until the hash processor is ready (no block is being processed), that is wait for DINIS = 1 in HASH_SR. If DCIS is also set in HASH_SR, the hash result is available and the context swapping is useless. Else go to step 5.
5. Save HASH_IMR, HASH_STR and HASH_CR registers. Also save a number of HASH_CSRx registers depending on the SHA algorithm that is used:
   - When SHA-1 or SHA2-256 is selected, save HASH_CSR0 to HASH_CSR37, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing.
   - When SHA2-384 or SHA2-512 is selected, save HASH_CSR0 to HASH_CSR90, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.

To resume the processing of a message using DMA, proceed as follows:

1. Reconfigure the DMA controller so that it proceeds with the transfer of the message up to the end if it is not interrupted again.
2. Program the values saved in memory to HASH_IMR, HASH_STR and HASH_CR registers.
3. Initialize the hash processor by setting the INIT bit in the HASH_CR register.
4. Program the values saved in memory to the HASH_CSRx registers.
5. Restart the processing from the point where it was interrupted by setting the DMAE bit.

32.4.9 HASH DMA interface

The HASH supports both single and fixed DMA burst transfers of four words.

The hash processor provides an interface to connect to the DMA controller. This DMA can be used to write data to the HASH by setting the DMAE bit in the HASH_CR register. When this bit is set, the HASH initiates a DMA request each time a block has to be written to the HASH_DIN register.

Once four 32-bit words have been received, the HASH automatically triggers a new request to the DMA. For more information, refer to *Section 32.4.5: Message digest computing*.

Before starting the DMA transfer, the software must program the number of valid bits in the last word that is copied into HASH_DIN register. This is done by writing in HASH_STR register the following value:

`NBLW = Len(Message)% 32` where "**`x%32`**" gives the remainder of x divided by 32.

The DMAS bit of the HASH_SR register provides information on the DMA interface activity. This bit is set with DMAE and cleared when DMAE is cleared and no DMA transfer is ongoing.

Note: No interrupt is associated to DMAS bit.

When MDMAT is set, the size of the transfer must be a multiple of four words.

32.4.10 HASH error management

No error flags are generated by the hash processor.

32.4.11 HASH processing time

Table 322 summarizes the time required to process an intermediate block for each mode of operation.

Table 322. Processing time (in clock cycle)

Mode of operation	Block size (in bytes)	FIFO load(1)	Computation phase	Total
SHA-1	64	16	66	**82**
SHA2-224		16	50	**66**
SHA2-256				
SHA2-384	128	32	66	**98**
SHA2-512(2)				

1. Add the time required to load the block into the processor.
2. SHA2-512 includes SHA2-512/224 and SHA2-512/256 modes.

The time required to process the last block of a message (or of a key in HMAC) can be longer. This time depends on the length of the last block and the size of the key (in HMAC mode).

Compared to the processing of an intermediate block, it can be increased by the factor below:

- **1 to 2.5** for a hash message
- **~2.5** for an HMAC input-key
- **1 to 2.5** for an HMAC message
- **~2.5** for an HMAC output key in case of a short key
- **3.5 to 5** for an HMAC output key in case of a long key

32.5 HASH interrupts

Two individual maskable interrupt sources are generated by the hash processor to signal the following events:

- Digest calculation completion (DCIS)
- Data input buffer ready (DINIS)

Both interrupt sources are connected to the same global interrupt request signal (hash_it), which is in turn connected to the device interrupt controller. Each interrupt source can individually be enabled or disabled by changing the mask bits in the HASH_IMR register. Setting the appropriate mask bit enables the interrupt.

The status of each maskable interrupt source can be read from the HASH_SR register. *Table 323* gives a summary of the available features.

Table 323. HASH interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
HASH	Digest computation completed	DCIS	DCIE	Clear DCIS or set INIT
	Data input buffer ready to get a new block	DINIS	DINIE	Clear DINIS or write to HASH_DIN

32.6 HASH registers

The hash core is associated with several control and status registers and several message digest registers. All these registers are accessible through 32-bit word accesses only, else an AHB error is generated.

32.6.1 HASH control register (HASH_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[3:0]				LKEY
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MDMAT	DINNE	NBW[3:0]				Res.	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
		rw	r	r	r	r	r		rw	rw	rw	rw	w		

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **ALGO[3:0]**: Algorithm selection

These bits select the hash algorithm. This selection is only taken into account when the INIT bit is set. Changing this bitfield during a computation has no effect

When the ALGO bitfield is updated and INIT bit is set, NBWE in HASH_SR is automatically updated to 0x11.

0000: SHA-1

0001: reserved

0010: SHA2-224

0011: SHA2-256

1100: SHA2-384

1101: SHA2-512/224

1110: SHA2-512/256

1111: SHA2-512

Bit 16 **LKEY**: Long key selection

The application must set this bit if the HMAC key is greater than the block size corresponding to the hash algorithm (see *Table 320: Information on supported hash algorithms* for details). For example the block size is 64 bytes for SHA2-256.

This selection is only taken into account when the INIT and MODE bits are set (HMAC mode selected). Changing this bit during a computation has no effect.

0: HMAC key is shorter or equal to the block size (short key). The actual key value written in HASH_DIN is used during the HMAC computation.

1: HMAC key is longer than the block size (long key). The hash of the key is used instead of the real key during the HMAC computation.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **MDMAT**: Multiple DMA transfers

This bit is set when hashing large files when multiple DMA transfers are needed.

0: DCAL is automatically set at the end of a DMA transfer.

1: DCAL is not automatically set at the end of a DMA transfer.

Bit 12 **DINNE**: DIN not empty

Refer to DINNE bit of HASH_SR for a description of DINNE bit.

This bit is read-only.

Bits 11:8 **NBW[3:0]:** Number of words already pushed

Refer to NBWP[3:0] bitfield of HASH_SR for a description of NBW[3:0] bitfield.

This bitfield is read-only.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **MODE:** Mode selection

This bit selects the normal or the keyed HMAC mode for the selected algorithm. This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

0: Hash mode selected

1: HMAC mode selected. LKEY bit must be set if the key being used is longer than the algorithm block size.

Bits 5:4 **DATATYPE[1:0]:** Data type selection

This bitfield defines the format of the data entered into the HASH_DIN register:
00: 32-bit data. The data written into HASH_DIN are directly used by the HASH processing, without reordering.
01: 16-bit data or half-word. The data written into HASH_DIN are considered as two half-words, and are swapped before being used by the HASH processing.
10: 8-bit data or bytes. The data written into HASH_DIN are considered as four bytes, and are swapped before being used by the HASH processing.
11: bit data or bit string. The data written into HASH_DIN are considered as 32 bits (1st bit of the string at position 0), and are swapped before being used by the HASH processing (1st bit of the string at position 31).

Bit 3 **DMAE:** DMA enable

Setting this bit enables DMA transfers. After this bit is set, it is cleared by hardware while the last data of the message is written into the hash processor.
Clearing this bit while a DMA transfer is ongoing does not abort the current transfer. Instead, the DMA interface of the HASH remains internally enabled until the transfer is complete or INIT is set.
Setting INIT bit does not clear DMAE bit.
0: DMA transfers disabled
1: DMA transfers enabled. A DMA request is sent as soon as the hash core is ready to receive data.

Bit 2 **INIT:** Initialize message digest calculation

This is a write-only bit always returning zero upon reading. Setting this bit resets the hash processor core, so that HASH is ready to compute the message digest of a new message. Clearing this bit has no effect.

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

32.6.2 HASH data input register (HASH_DIN)

Address offset: 0x004

Reset value: 0x0000 0000

HASH_DIN is the data input register. It is 32-bit wide. This register is used to enter the message by blocks defined by the hash algorithm (see *Table 320: Information on supported hash algorithms* for details). For example the block size is 64 bytes for SHA2-256.

When the HASH_DIN register is programmed, the value presented on the AHB bus is ‘pushed’ into the hash core and the register takes the new value presented on the AHB bus. To get a correct message format, the DATATYPE bits must have been previously configured in the HASH_CR register.

When a complete block has been written to the HASH_DIN register, an intermediate digest calculation is launched:

- by writing first data of the next block into the HASH_DIN register, if the DMA is not used
- automatically, if the DMA is used

When the last block has been written to the HASH_DIN register, the final digest calculation (including padding) is launched by setting the DCAL bit in the HASH_STR register (final digest calculation). This operation is automatic if the DMA is used and MDMAT bit is cleared.

Reading the HASH_DIN register returns zeros.

Note: When HASH is busy, a write access to the HASH_DIN register might stall the AHB bus if the digest calculation (intermediate or final) is not complete.

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

Bits 31:0 **DATAIN[31:0]:** Data input

Writing this register pushes the current register content into the FIFO, and the register takes the new value presented on the AHB bus.

Do not write to this register when BUSY is set.

Reading this register returns zeros.

32.6.3 HASH start register (HASH_STR)

Address offset: 0x008

Reset value: 0x0000 0000

The HASH_STR register has two functions:

- It is used to define the number of valid bits in the last word of the message entered in the hash processor (that is the number of valid least significant bits in the last data written to the HASH_DIN register)
- It is used to start the processing of the last block in the message by setting the DCAL bit.

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **DCAL:** Digest calculation

This is a write-only bit always returning zero upon reading. Setting this bit starts the message padding using the previously written value of NBLW, and starts the calculation of the final message digest with all the data words written to the input FIFO since the INIT bit was last set.

Do not set DCAL when BUSY is set.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **NBLW[4:0]:** Number of valid bits in the last word

When the last word of the message bit string is written to HASH_DIN register, the hash processor takes only the valid bits, specified as below, after internal data swapping.

The above mechanism is valid only if DCAL = 0. If NBLW bits are written while DCAL is set, the NBLW bitfield remains unchanged. In other words it is not possible to configure NBLW and set DCAL at the same time.

Reading NBLW bits returns the last value written to NBLW.

0x00: All the 32 bits of the last data written are valid message bits, that is M[31:0]

0x01: Only one bit of the last data written (after swapping) is valid, that is M[0]

0x02: Only two bits of the last data written (after swapping) are valid, that is M[1:0]

0x03: Only three bits of the last data written (after swapping) are valid that is M[2:0]

...

0x1F: Only 31 bits of the last data written (after swapping) are valid that is M[30:0]

Note:

32.6.4 HASH digest registers

These registers contain the message digest result defined as follows:

- HASH_HR0, HASH_HR1, HASH_HR2, HASH_HR3 and HASH_HR4 registers return the SHA-1 digest result.
- HASH_HR0 to HASH_HR6 registers return the SHA2-224, SHA2-512/224 digest result.
- HASH_HR0 to HASH_HR7 registers return the SHA2-256, SHA2-512/256 digest result.
- HASH_HR0 to HASH_HR11 registers return the SHA2-384 digest result.
- HASH_HR0 to HASH_HR15 registers return the SHA2-512 digest result

In all cases, the digest most significant bit is stored in HASH_HR0[31], and unused HASH_HRx registers reads as zero.

If a read access to one of these registers is performed while the hash core is calculating an intermediate digest or a final message digest (DCIS bit equals 0), then the read operation returns zeros.

Note: When starting a digest computation for a new message (by setting the INIT bit), HASH_HRx registers are forced to their reset values.

HASH_HR0 to HASH_HR4 registers can be accessed through two different addresses (register aliasing).

HASH aliased digest register x (HASH_HRAx)

Address offset: 0x00C + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

The content of the HASH_HRAx registers is identical to the one of the HASH_HRx registers located at address offset 0x310.

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

Bits 31:0 **Hx[31:0]:** Hash data x
Refer to *Section 32.6.4: HASH digest registers* introduction.

HASH digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

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

Bits 31:0 **Hx[31:0]:** Hash data x
Refer to *Section 32.6.4: HASH digest registers* introduction.

HASH supplementary digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * x, (x = 5 to 15)

Reset value: 0x0000 0000

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

Bits 31:0 **Hx[31:0]:** Hash data x
Refer to *Section 32.6.4: HASH digest registers* introduction.

32.6.5 HASH interrupt enable register (HASH_IMR)

Address offset: 0x020

Reset value: 0x0000 0000

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

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

Bit 1 **DCIE:** Digest calculation completion interrupt enable
0: Digest calculation completion interrupt disabled
1: Digest calculation completion interrupt enabled.

Bit 0 **DINIE:** Data input interrupt enable
0: Data input interrupt disabled
1: Data input interrupt enabled

32.6.6 HASH status register (HASH_SR)

Address offset: 0x024

Reset value: 0x0011 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBWE[4:0]				
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DINNE	Res.	NBWP[4:0]					Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
r		r	r	r	r	r						r	r	rc_w0	rc_w0

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **NBWE[4:0]:** Number of words expected
This bitfield reflects the number of words in the message that must be pushed into the FIFO to trigger a partial computation. NBWE is decremented by 1 when a write access is performed to the HASH_DIN register.
NBWE is set to the expected block size +1 in words (0x11) when INIT bit is set in HASH_CR. It is set to the expected block size (0x10) when the partial digest calculation ends.

Bit 15 **DINNE:** DIN not empty
This bit is set when the HASH_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is set.
0: No data are present in the data input buffer
1: The input buffer contains at least one word of data

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **NBWP[4:0]:** Number of words already pushed

This bitfield is the exact number of words in the message that have already been pushed into the FIFO. NBWP is incremented by 1 when a write access is performed to the HASH_DIN register.

When a digest calculation starts, NBWP is updated to NBWP- block size (in words), and NBWP goes to zero when the INIT bit is set.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy bit

0: No block is currently being processed
1: The hash core is processing a block of data

Bit 2 **DMAS:** DMA Status

This bit provides information on the DMA interface activity. It is set with DMAE and cleared when DMAE = 0 and no DMA transfer is ongoing. No interrupt is associated with this bit.

0: DMA interface is disabled (DMAE = 0) and no transfer is ongoing
1: DMA interface is enabled (DMAE = 1) or a transfer is ongoing

Bit 1 **DCIS:** Digest calculation completion interrupt status

This bit is set by hardware when a digest becomes ready (the whole message has been processed). It is cleared by writing it to 0 or by setting the INIT bit in the HASH_CR register.

0: No digest available in the HASH_HRx registers (zeros are returned)

1: Digest calculation complete, a digest is available in the HASH_HRx registers. An interrupt is generated if the DCIE bit is set in the HASH_IMR register.

Bit 0 **DINIS:** Data input interrupt status

This bit is set by hardware when the FIFO is ready to get a new block (16 locations are free). It is cleared by writing it to 0 or by writing the HASH_DIN register.

When DINIS = 0, HASH_CSRx registers reads as zero.

0: Less than 16 locations are free in the input buffer
1: A new block can be entered into the input buffer. An interrupt is generated if the DINIE bit is set in the HASH_IMR register.

32.6.7 HASH context swap registers

These registers contain the complete internal register states of the hash processor. They are useful when a suspend/resume operation has to be performed because a high-priority task needs to use the hash processor while it is already used by another task.

When such an event occurs, the HASH_CSRx registers have to be read and the read values have to be saved in the system memory space. Then the hash processor can be used by the preemptive task. When the hash computation is complete, the saved context can be read from memory and written back into the HASH_CSRx registers.

HASH_CSRx registers can be read only when DINIS equals to 1, otherwise zeros are returned.

HASH context swap register x (HASH_CSRx)

Address offset: 0x0F8 + 0x4 * x, (x = 0 to 102)

Reset value: 0x0022 0002 (HASH_CSR0)

Reset value: 0x0000 0000 (others)

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

Bits 31:0 **CSx[31:0]:** Context swap x

Refer to *Section 32.6.7: HASH context swap registers* introduction.

32.6.8 HASH register map

Table 324. HASH1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	HASH_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[3:2]		ALGO[1:0]		LKEY	Res.	Res.	MDMAT	DINNE	NBW[3:0]				Res.	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
	Reset value												0	0	0	0	0			0	0	0	0	0	0		0	0	0	0	0		
0x004	HASH_DIN	DATAIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	HASH_STR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
	Reset value																								0				0	0	0	0	0
0x00C + 0x4 * x, (x = 0 to 4) Last address: 0x01C	HASH_HRAx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	HASH_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
	Reset value																															0	0
0x024	HASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		NBWE[4:0]					DINNE		NBWP[4:0]					Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
	Reset value												1	0	0	0	1	0		0	0	0	0	0						0	0	0	1
Reserved																																	
0x0F8	HASH_CSR0	CSR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 324. HASH1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0FC	HASH_CSR1	CSR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	HASH_CSR2	CSR2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x1CC	HASH_CSR53	CSR53[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x290	HASH_CSR102	CSR102[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reserved																																	
0x310 + 0x4 * x, (x = 0 to 15) Last address: 0x34C	HASH_HRx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

33 High-resolution timer (HRTIM)

33.1 Introduction

The high-resolution timer can generate up to 12 digital signals with highly accurate timings. It is primarily intended to drive power conversion systems such as switch mode power supplies or lighting systems, but can be of general purpose usage, whenever a very fine timing resolution is expected.

Its modular architecture is used to generate either independent or coupled waveforms. The wave-shape is defined by self-contained timings (using counters and compare units) and a broad range of external events, such as analog or digital feedback and synchronization signals. This is used to produce a large variety of control signals (PWM, phase-shifted, constant Ton,...) and address most of the conversion topologies.

For control and monitoring purposes, the timer also has timing measure capabilities and links to built-in ADC and DAC converters. Last, it features a light-load management mode and is able to handle various fault schemes for safe shut-down purposes.

Caution: Unlike in other STM32 microcontrollers dedicated for digital power conversion, this HRTIM is not capable of subnanosecond resolution:

- The resolution is directly related to the input clock of the timer (no clock multiplication factor)
- It does not include the DLL feature and calibration logic

However, it supports all the features of the subnanosecond capable HRTIM and is suitable for a wide range of applications requiring an extremely versatile PWM generation with slightly relaxed resolution needs.

33.2 Main features

- Multiple timing units
  - Down to 1.9 ns resolution
  - Duty cycle, frequency, phase shift, and pulse width (triggered one-pulse mode) can be adjusted with full resolution, on all outputs
  - 6 16-bit timing units (each one with an independent counter and four compare units)
  - 12 outputs that can be controlled by any timing unit, up to 32 set/reset sources per channel
  - Modular architecture to address either multiple independent converters with one or two switches or few large multiswitch topologies
- Up to 16 external events, available for any timing unit
  - Suitable for peak current control, zero voltage, and zero current detection (ZVS and ZCS operation) and counter reset (variable frequency operation)
  - 4 sources per external events (digital inputs, built-in comparators outputs, ADC's analog watchdogs, and general purpose timer's trigger outputs)
  - Programmable polarity and edge sensitivity
  - 8 events with a fast asynchronous mode
  - 8 events with a programmable digital filter
  - Spurious events filtering with blanking and windowing modes
- Direct connections to built-in analog peripherals
  - 10 triggers to ADC converters
  - 9 triggers for DAC converters, among which 6 with slope compensation capabilities
  - Comparator outputs (including filtering and blanking features)
- Versatile protection scheme
  - Suitable for overcurrent, short-circuit, and over-voltage/under-voltage protection
  - 8 fault inputs can be combined and associated with any timing unit
  - 3 sources per fault channel (digital inputs, built-in comparators outputs, and external events)
  - Programmable polarity, edge sensitivity, and programmable digital filter
  - Dedicated delayed protections for resonant converters
- Multiple HRTIM instances can be synchronized with external synchronization inputs/outputs
- Versatile output stage
  - High-resolution deadtime insertion (down to 1.9 ns)
  - Programmable output polarity
  - Chopper mode
- Burst mode controller to handle light-load operation synchronously on multiple converters
- 8 interrupt vectors, each one with up to 14 sources
- 7 DMA requests with up to 14 sources, with a burst mode for multiple registers update

33.3 Functional description

33.3.1 General description

The HRTIM can be partitioned into several subentities:

- The master timer
- The timing units (timer A to timer F)
- The output stage
- The burst mode controller
- An external event and fault signal conditioning logic that is shared by all timers
- The system interface

The master timer is based on a 16-bit up counter. It can set/reset any of the 12 outputs via four compare units and it provides synchronization signals to the six timer units. Its main purpose is to have the timer units controlled by a unique source. An interleaved buck converter is a typical application example where the master timer manages the phase-shifts between the multiple units.

The timer units are working either independently or coupled with the other timers including the master timer. Each timer contains the controls for two outputs. The outputs set/reset events are triggered either by the timing units compare registers or by events coming from the master timer, from the other timers, or from external events.

The output stage has several duties

- Addition of deadtime when the two outputs are configured in complementary PWM mode
- Addition of a carrier frequency on top of the modulating signal
- Management of fault events, by asynchronously asserting the outputs to a predefined safe level

The burst mode controller can take over the control of one or multiple timers in case of light-load operation. The burst length and period can be programmed, as well as the idle state of the outputs.

The external event and fault signal conditioning logic includes:

- The input selection MUXes (for instance for selecting a digital input or an on-chip source for a given external event channel)
- Polarity and edge-sensitivity programming
- Digital filtering (for 5 channels out of 12)

The system interface allows the HRTIM to interact with the rest of the MCU:

- Interrupt requests to the CPU
- DMA controller for automatic accesses to/from the memories, including an HRTIM specific burst mode
- Triggers for the ADC and DAC converters

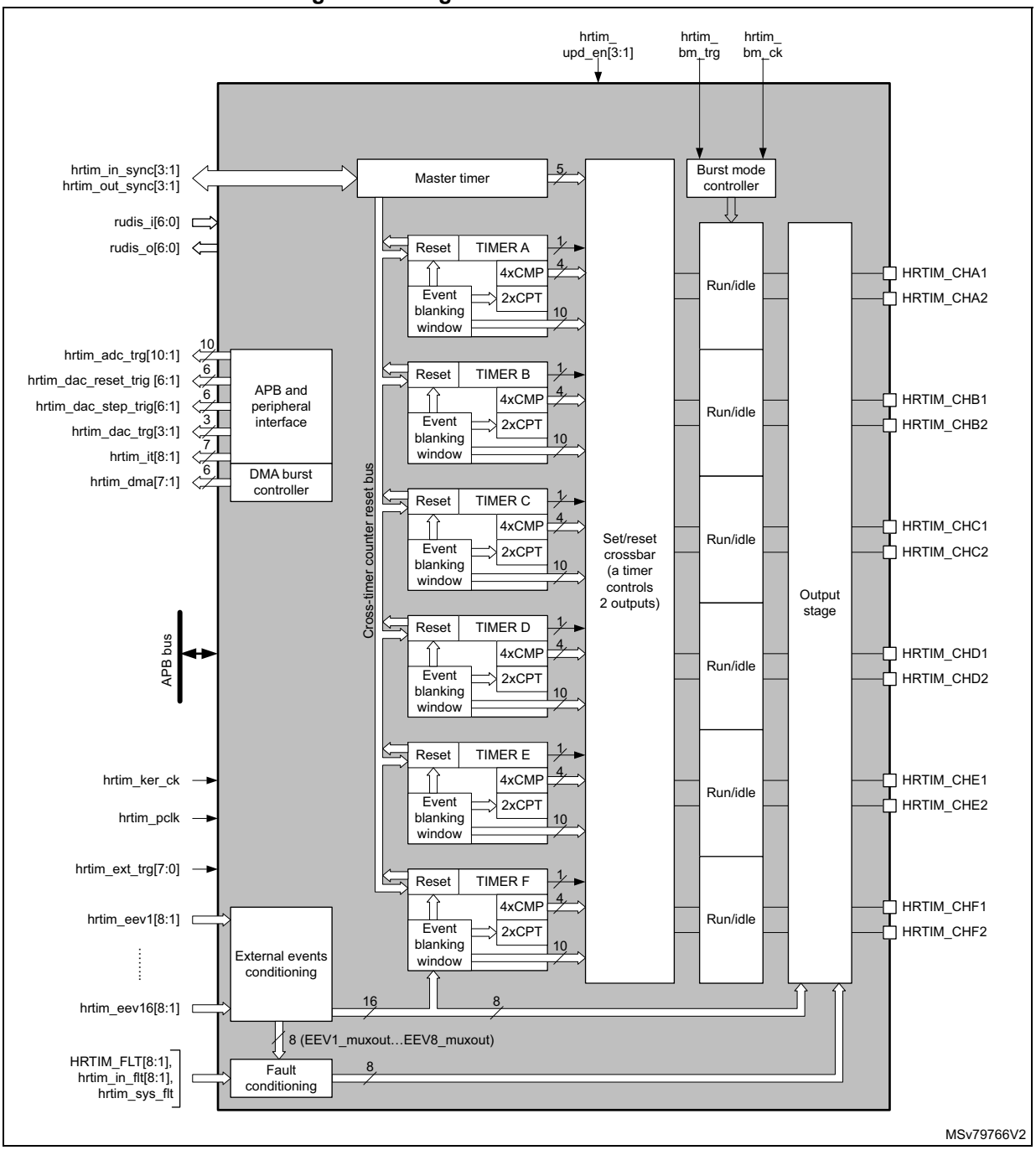
The HRTIM registers are split into eight groups:

- Master timer registers
- Timer A to timer F registers
- Common registers for features shared by all timer units

Note: As a writing convention, references to the six timing units in the text and in the registers are generalized using the "x" letter, where x can be any value from A to F.

The block diagram of the timer is shown in *Figure 422*.

Figure 422. High-resolution timer overview



33.3.2 HRTIM pins and internal signals

The tables in this section summarize the HRTIM inputs and outputs, both on-chip and off-chip.

Table 325. HRTIM inputs/outputs summary

Signal name	Signal type	Description
HRTIM_CHA1, HRTIM_CHA2, HRTIM_CHB1, HRTIM_CHB2, HRTIM_CHC1, HRTIM_CHC2, HRTIM_CHD1, HRTIM_CHD2, HRTIM_CHE1, HRTIM_CHE2, HRTIM_CHF1, HRTIM_CHF2	Outputs	Main HRTIM timer outputs. They can be coupled by pairs (HRTIM_CHx1 & HRTIM_CHx2) with deadtime insertion or work independently.
HRTIM_FLT[8:1] hrtim_in_flt[8:1]	Input	Fault inputs: immediately disable the HRTIM outputs when asserted (8 on-chip inputs and 8 off-chip HRTIM_FLTx inputs).
hrtim_sys_flt	Input	System fault gathering MCU internal fault events (clock security system, SRAM, D-cache and flash memory double error detection, Cortex®-M7 lockup (HardFault), PVD output.
hrtim_in_sync[3:1]	Input	Synchronization inputs to synchronize the whole HRTIM with other internal or external timer resources: hrtim_in_sync1: the source is another internal HRTIM (via on-chip interconnect) hrtim_in_sync2: the source is an internal timer output hrtim_in_sync3: the source is HRTIM_SCIN input pins
hrtim_out_sync[2:1]	Output	The purpose of this output is to cascade or synchronize several HRTIM instances, either on-chip or off-chip: hrtim_out_sync1: the destination is an on-chip HRTIM (via internal interconnect) hrtim_out_sync2: the destination is an off-chip HRTIM or peripheral (via HRTIM_SCOUT output pins)

Table 325. HRTIM inputs/outputs summary (continued)

Signal name	Signal type	Description
hrtim_eev1[8:1]	Input	External events. Each of the 16 events can be selected among 8 sources, either on-chip (from other built-in peripherals: comparator, ADC analog watchdog, TIMx timers, trigger outputs) or off-chip (HRTIM_EEVx input pins).
hrtim_eev2[8:1]		
hrtim_eev3[8:1]		
hrtim_eev4[8:1]		
hrtim_eev5[8:1]		
hrtim_eev6[8:1]		
hrtim_eev7[8:1]		
hrtim_eev8[8:1]		
hrtim_eev9[8:1]		
hrtim_eev10[8:1]		
hrtim_eev11[8:1]		
hrtim_eev12[8:1]		
hrtim_eev13[8:1]		
hrtim_eev14[8:1]		
hrtim_eev15[8:1]		
hrtim_eev16[8:1]		
hrtim_ext_trg[7:0]	Input	External triggers, for crossbar override and extra faults inputs
hrtim_upd_en[3:1]	Input	A pulse on the update enable inputs hrtim_upd_end[3:1] (on-chip interconnect) triggers the transfer from shadow to active registers
hrtim_bm_trg	Input	A pulse on this input triggers a burst mode entry.
hrtim_bm_ck[4:1]	Input	Burst mode clock (on-chip interconnect)
hrtim_adc_trg[10:1]	Output	ADC start of conversion triggers. The hrtim_adc_trg1 .. hrtim_adc_trg10 outputs are here-after referred to as ADC trigger 1 to ADC trigger 10.
hrtim_dac_trg[3:1]	Output	DAC conversion update triggers
hrtim_dac_reset_trg[6:1] hrtim_dac_step_trg[6:1]	Output	Dual channel DAC triggers
rudis_i[6:0]	Input	Remote update disable inputs (receiver)
rudis_o[6:0]	Output	Remote update disable outputs (requester)
hrtim_it[8:1]	Output	Interrupt requests
hrtim_dma[7:1]	Output	DMA requests
hrtim_pclk	Input	APB clock
hrtim_ker_ck	-	HRTIM kernel clock

Table 326. HRTIM1/HRTIM2 external events mapping and associated features

External event channel	Fast mode	Digital filter	Balanced fault timer A,B,C	Balanced fault timer D,E,F	Source (EExSRC[2:0])							
					Src1 (000)	Src 2 (001)	Src3 (010)	Src4 (011)	Src5 (100)	Src 6 (101)	Src7 (110)	Src8 (111)
hrtim_eev1[8:1]	Yes	-	-	-	HRTIM_EEV1(1)	comp2_out	tim1_trgo	adc1_awd1	adc4_awd1	-	-	-
hrtim_eev2[8:1]	Yes	-	-	-	HRTIM_EEV2(1)	comp4_out	tim2_trgo	adc1_awd2	adc4_awd2	-	-	-
hrtim_eev3[8:1]	Yes	-	-	-	HRTIM_EEV3(1)	comp6_out	tim3_trgo	adc1_awd3	adc4_awd3	-	-	-
hrtim_eev4[8:1]	Yes	-	-	-	HRTIM_EEV4(1)	comp1_out	comp5_out	adc2_awd1	adc3_awd1	-	-	-
hrtim_eev5[8:1]	Yes	-	-	-	HRTIM_EEV5(1)	comp3_out	comp7_out	adc2_awd2	adc3_awd2	-	-	-
hrtim_eev6[8:1]	-	Yes	Yes	-	HRTIM_EEV6(1)	comp2_out	comp1_out	adc2_awd3	adc3_awd3	-	-	-
hrtim_eev7[8:1]	-	Yes	Yes	-	HRTIM_EEV7(1)	comp4_out	comp7_out	adc3_awd1	adc4_awd1	-	-	-
hrtim_eev8[8:1]	-	Yes	-	Yes	HRTIM_EEV8(1)	comp6_out	comp3_out	adc3_awd2	adc4_awd2	-	-	-
hrtim_eev9[8:1]	-	Yes	-	Yes	HRTIM_EEV9(1)	comp5_out	comp8_out	comp4_out	-	-	-	-
hrtim_eev10[8:1]	-	Yes	-	-	HRTIM_EEV10(1)	comp7_out	tim6_trgo	adc3_awd3	adc4_awd3	-	-	-
hrtim_eev11[8:1]	Yes	-	Yes	-	HRTIM_EEV11(1)	comp7_out	tim9_trgo	adc2_awd1	-	-	-	-
hrtim_eev12[8:1]	Yes	-	-	Yes	HRTIM_EEV12(1)	comp8_out	tim12_trgo	adc2_awd2	-	-	-	-
hrtim_eev13[8:1]	Yes	-	-	Yes	HRTIM_EEV13(1)	comp5_out	tim15_trgo	adc2_awd3	-	-	-	-

Table 326. HRTIM1/HRTIM2 external events mapping and associated features (continued)

External event channel	Fast mode	Digital filter	Balanced fault timer A,B,C	Balanced fault timer D,E,F	Source (EExSRC[2:0])							
					Src1 (000)	Src 2 (001)	Src3 (010)	Src4 (011)	Src5 (100)	Src 6 (101)	Src7 (110)	Src8 (111)
hrtim_eev14[8:1]	-	Yes	-	-	HRTIM_EEV14(1)	comp8_out	tim1_trgo	adc1_awd1	-	-	-	-
hrtim_eev15[8:1]	-	Yes	-	-	HRTIM_EEV15(1)	comp1-out	tim8_trgo	adc1_awd2	-	-	-	-
hrtim_eev16[8:1]	-	Yes	Yes	-	HRTIM_EEV16(1)	comp3_out	tim20_trgo	adc1_awd3	-	-	-	-

1. Refer to product datasheet for alternate function mapping

Table 327. HRTIM1/HRTIM2 update enable inputs and sources

hrtim_upd_en[3:1]	Update source
hrtim_upd_en1	tim10_oc1
hrtim_upd_en2	tim11_oc1
hrtim_upd_en3	tim6_trgo

Table 328. HRTIM1/HRTIM2 Burst mode clock sources

hrtim_bm_ck[4:1]	BMCLK[3:0]	Clock source
hrtim_bm_ck1	0110	tim12_oc1
hrtim_bm_ck2	0111	tim13_oc1
hrtim_bm_ck3	1000	tim7_trgo
hrtim_bm_ck4	1001	tim10_trgo

Table 329. HRTIM1/HRTIM2 Burst mode trigger source

-	trigger source
hrtim_bm_trg	tim6_trgo

Table 330. HRTIM1/HRTIM2 internal synchronization sources

-	HRTIM1	HRTIM2
hrtim_in_sync1	hrtim2_out_sync1	hrtim1_out_sync1
hrtim_in_sync2	tim1_trgo	
hrtim_in_sync3	HRTIM1_SCIN	HRTIM2_SCIN

Table 331. HRTIM1/HRTIM2 remote update disable interconnect

-	HRTIM1	HRTIM2
rudis_i0 (Master timer)	hrtim2_rudis_o0	hrtim1_rudis_o0
rudis_i1 (Timer A)	hrtim2_rudis_o1	hrtim1_rudis_o1
rudis_i2 (Timer B)	hrtim2_rudis_o2	hrtim1_rudis_o2
rudis_i3 (Timer C)	hrtim2_rudis_o3	hrtim1_rudis_o3
rudis_i4 (Timer D)	hrtim2_rudis_o4	hrtim1_rudis_o4

Table 331. HRTIM1/HRTIM2 remote update disable interconnect (continued)

-	HRTIM1	HRTIM2
rudis_i5 (Timer E)	hrtim2_rudis_o5	hrtim1_rudis_o5
rudis_i6 (Timer F)	hrtim2_rudis_o6	hrtim1_rudis_o6

Table 332. HRTIM1/HRTIM2 system fault sources selection

HRTIM_OR register	hrtim1_sys_flt	hrtim2_sys_flt
HSPBKGSEL = 0	Cortex-M7 lockup PVD low threshold CSS Double ECC errors in AXISRAM1-4, ITCM, DTCM, data cache, backup RAM, and flash memory.	
HSPBKGSEL = 1	Cortex-M7 lockup PVD low threshold CSS Double ECC errors in AXISRAM1-4, ITCM, DTCM, data cache, backup RAM, and flash memory hsp1_break_out0	Cortex-M7 lockup PVD low threshold CSS Double ECC errors in AXISRAM1-4, ITCM, DTCM, data cache, backup RAM, and flash memory hsp1_break_out2

Table 333. HRTIM1/HRTIM2 fault inputs

Fault channel	External Input FLTxSRC[1:0] = 00 (HRTIM_FLT[8:1])	On-chip source FLTxSRC[1:0] = 01 (hrtim_flt_in[8:1])	External Input FLTxSRC[1:0] = 10 (HRTIM internal(1))	On-chip source FLTxSRC[1:0] = 11 hrtim_ext_trg[7:0]
FAULT 1	HRTIM_FLT1(2)	comp2_out	EEV1_mux_out	-
FAULT 2	HRTIM_FLT2(2)	comp4_out	EEV2_mux_out	-
FAULT 3	HRTIM_FLT3(2)	comp6_out	EEV3_mux_out	-
FAULT 4	HRTIM_FLT4(2)	comp1_out	EEV4_mux_out	-
FAULT 5	HRTIM_FLT5(2)	comp3_out	EEV5_mux_out	-
FAULT 6	HRTIM_FLT6(2)	comp5_out	EEV6_mux_out	-
FAULT 7	HRTIM_FLT7(2)	comp7_out	EEV7_mux_out	-
FAULT 8	HRTIM_FLT8(2)	comp8_out	EEV8_mux_out	-

1. Refer to *Section 33.3.17: Fault protection* for details
2. Refer to product datasheet for alternate function mapping

Table 334. HRTIM1/HRTIM2 external trigger inputs

-	HRTIM1	HRTIM2
hrtim_ext_trg0	-	-
hrtim_ext_trg1	-	-
hrtim_ext_trg2	-	-
hrtim_ext_trg3	-	-
hrtim_ext_trg4	-	-
hrtim_ext_trg5	-	-
hrtim_ext_trg6	-	-
hrtim_ext_trg7	-	-

Table 335. HRTIM1/HRTIM2 DAC triggers connections

HRTIM DAC triggers	DAC1	DAC2	DAC3	DAC4
hrtim_dac_trg1	Yes	Yes	Yes	Yes
hrtim_dac_trig2	Yes	Yes	Yes	Yes
hrtim_dac_trig3	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg1 hrtim_dac_step_trg1	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg2 hrtim_dac_step_trg2	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg3 hrtim_dac_step_trg3	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg4 hrtim_dac_step_trg4	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg5 hrtim_dac_step_trg5	Yes	Yes	Yes	Yes
hrtim_dac_reset_trg6 hrtim_dac_step_trg6	Yes	Yes	Yes	Yes

33.3.3 Clocks

The HRTIM must be supplied by the highest possible kernel clock frequency hrtim_ker_ck (here-after referred as f_{HRTIM}) to offer full resolution. All clocks present in the HRTIM are derived from this reference clock.

Definition of terms

f_{HRTIM}: Main HRTIM clock (hrtim_ker_ck). All subsequent clocks are derived and synchronous with this source.

f_{DTG}: Deadtime generator clock. For convenience, only the t_{DTG} period (t_{DTG} = $1/f_{DTG}$) is used in this document.

f_{CHPFRQ}: Chopper stage clock source.

f_{1STPW}: Clock source defining the length of the initial pulse in chopper mode. For convenience, only the t_{1STPW} period ($t_{1STPW} = 1/f_{1STPW}$) is used in this document.

f_{BRST}: Burst mode controller counter clock.

$f_{SAMPLING}$: Clock needed to sample the fault or the external events inputs.

f_{FLTS}: Clock derived from f_{HRTIM} which is used as a source for $f_{SAMPLING}$ to filter fault events.

f_{EEVS}: Clock derived from f_{HRTIM} which is used as a source for $f_{SAMPLING}$ to filter external events.

Timer clock and prescaler

Each timer in the HRTIM has its own individual clock prescaler, which is used to adjust the timer resolution (see *Table 336*).

Table 336. Timer resolution and min. PWM frequency for f_{HRTIM} = 520 MHz

CKPSC[2:0] (1)	Prescaling ratio	$f_{COUNTER}$	Resolution	Min PWM frequency
101	1	520 MHz	1.9 ns	7.9 kHz
110	2	520/2 MHz = 210 MHz	3.8 ns	4 kHz
111	4	520/4 MHz = 105 MHz	7.6 ns	2 kHz

1. CKPSC[2:0] values from 000 to 100 are reserved.

The high-resolution is available for edge positioning, PWM period adjustment, and externally triggered pulse duration.

Initialization

At startup, it is mandatory to initialize first the prescaler bitfields before writing the compare and period registers. Once the timer is enabled (MCEN or TxCEN bit set in the HRTIM_MCR register), the prescaler cannot be modified.

When multiple timers are enabled, the prescalers are synchronized with the prescaler of the timer that was started first.

Warning: It is possible to have different prescaling ratios in the master and TIMA..E timers only if the counter and output behavior does not depend on other timers' information and signals. It is mandatory to configure identical prescaling ratios in these timers when one of the following events is propagated from one timing unit (or master timer) to another: output set/reset event, counter reset event, update event, external event filter, or capture triggers. Prescaler factors not equal yield to unpredictable results.

Deadtime generator clock

The deadtime prescaler is supplied by (f_{HRTIM} × 8) / $2^{(DTPRSC[2:0])}$, programmed with DTPRSC[2:0] bits in the HRTIM_DTxR register.

t_{DTG} ranges from 1.9 ns to 30.7 ns for f_{HRTIM} = 520 MHz.

Chopper stage clock

The chopper stage clock source f_{CHPFRQ} is derived from f_{HRTIM} with a division factor ranging from 16 to 256, so that 2.03 MHz ≤ f_{CHPFRQ} ≤ 32.5 MHz for f_{HRTIM} = 520 MHz.

t_{1STPW} is the length of the initial pulse in chopper mode, programmed with the STRPW[3:0] bits in the HRTIM_CHPxR register, as follows:

t_{1STPW} = (STRPW[3:0]+1) × 16 × t_{HRTIM}.

It uses f_{HRTIM} / 16 as clock source (32.5 MHz for f_{HRTIM}= 520 MHz).

Burst mode prescaler

The burst mode controller counter clock f_{BRST} can be supplied by several sources, among which one is derived from f_{HRTIM}.

In this case, f_{BRST} ranges from f_{HRTIM} to f_{HRTIM} / 32768 (15.9 kHz for f_{HRTIM} = 520 MHz).

Fault input sampling clock

The fault input noise rejection filter has a time constant defined with $f_{SAMPLING}$ which can be either f_{HRTIM} or f_{FLTS}.

f_{FLTS} is derived from f_{HRTIM} and ranges from 520 MHz to 65 MHz for f_{HRTIM} = 520 MHz.

External event input sampling clock

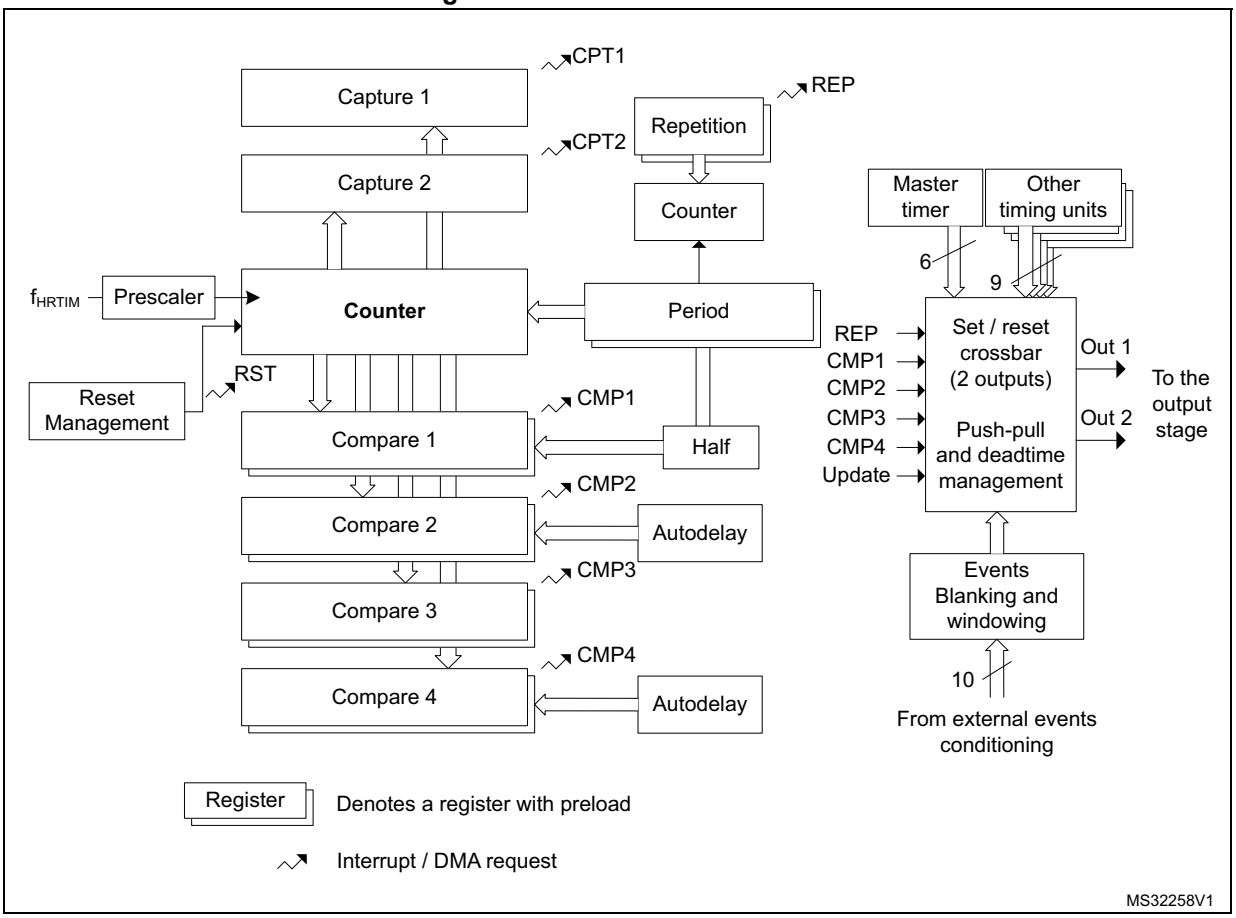
The fault input noise rejection filter has a time constant defined with $f_{SAMPLING}$ which can be either f_{HRTIM} or f_{EEVS}.

f_{EEVS} is derived from f_{HRTIM} and ranges from 520 MHz to 65 MHz for f_{HRTIM} = 520 MHz.

33.3.4 Timer A..F timing units

The HRTIM embeds six identical timing units made of a 16-bit upcounter with an autoreload mechanism to define the counting period, four compare, and two capture units, as per *Figure 423*. Each unit includes all control features for 2 outputs, so that it operates as a standalone timer.

Figure 423. Timer A..F overview



The period and compare values must be within a lower and an upper limit related to the high-resolution implementation and listed in *Table 337*:

- The minimum value must be greater than or equal to three periods of the f_{HRTIM} clock. The value 0x0000 can be written in CMP1 and CMP3 registers only, to skip a PWM pulse. See *Section : Null duty cycle exception case* for details
- The maximum value must be less than or equal to 0xFFFF - 1 periods of the f_{HRTIM} clock.

Table 337. Period and compare registers min and max values

CKPSC[2:0] value(1)	Min(2)	Max
≥ 5	0x0003	0xFFFD

1. CKPSC[2:0] values < 5 are reserved.
2. The value 0x0000 can be written in CMP1 and CMP3 registers only, to skip a PWM pulse. See *Section : Null duty cycle exception case* for details.

Note: A compare value greater than the period register value does not generate a compare match event.

Counter operating mode

Timer A..F operate in continuous (free-running) mode or in single-shot manner where counting is started by a reset event, using the CONT bit in the HRTIM_TIMxCR control register. An additional RETRIG bit allows you to select whether the single-shot operation is retriggerable or nonretriggerable. Details of the operation are summarized on *Table 338* and on *Figure 424* and *Figure 425*.

Table 338. Timer operating modes

CONT	RETRIG	Operating mode	Start / stop conditions Clocking and event generation
0	0	Single-shot Non-retriggerable	Setting the TxEN bit enables the timer but does not start the counter. A first reset event starts the counting and any subsequent reset is ignored until the counter reaches the PER value. The PER event is then generated and the counter is stopped. A reset event re-starts the counting operation from 0x0000.
0	1	Single-shot Retriggerable	Setting the TxEN bit enables the timer but does not start the counter. A reset event starts the counting if the counter is stopped, otherwise it clears the counter. When the counter reaches the PER value, the PER event is generated and the counter is stopped. A reset event re-starts the counting operation from 0x0000.
1	X	Continuous mode	Setting the TxEN bit enables the timer and starts the counter simultaneously. When the counter reaches the PER value, it rolls-over to 0x0000 and resumes counting. The counter can be reset at any time.

The TxEN bit can be cleared at any time to disable the timer and stop the counting.

Figure 424. Continuous timer operation

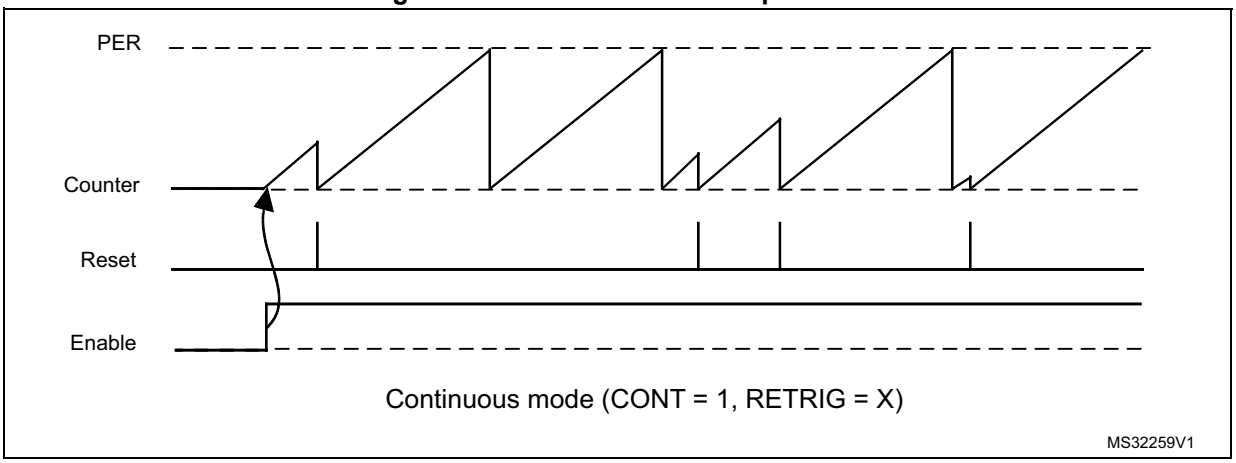
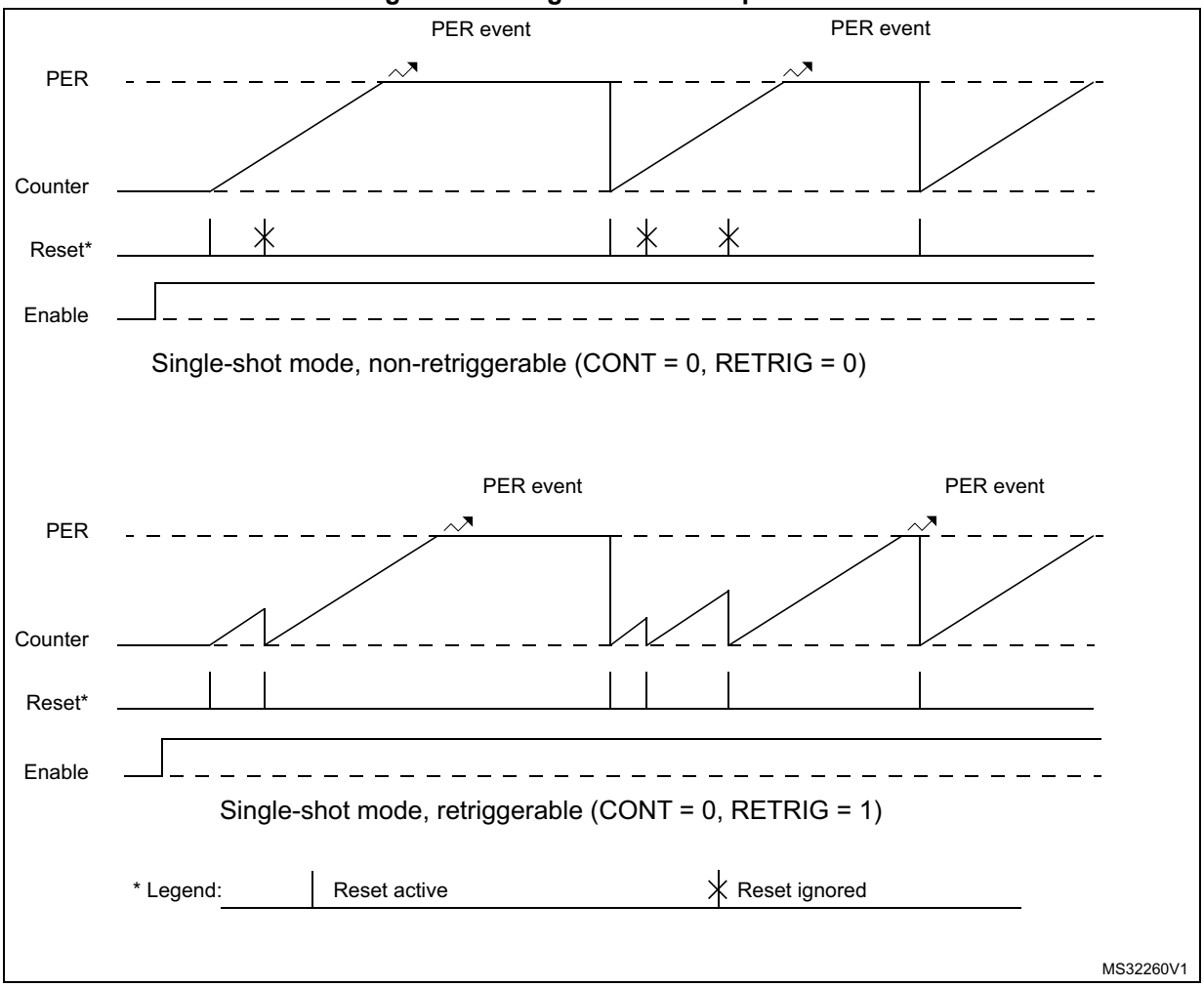


Figure 425. Single-shot timer operation



Roll-over event

A counter roll-over event is generated when the counter goes back to 0 after having reached the period value set in the HRTIM_PERxR register in continuous mode.

This event is used for multiple purposes in the HRTIM:

- To set/reset the outputs
- To trigger the register content update (transfer from preload to active)
- To trigger an IRQ or a DMA request
- To serve as a burst mode clock source or a burst start trigger
- As an ADC trigger
- To decrement the repetition counter

If the initial counter value is above the period value when the timer is started, or if a new period is set while the counter is already above this value, the counter is not reset: it overflows at the maximum period value and the repetition counter does not decrement.

Timer reset

The reset of the timing unit counter can be triggered by up to 30 events that can be selected simultaneously in the HRTIM_RSTxR register, among the following sources:

- The timing unit: compare 2, compare 4 and update (three events)
- The master timer: reset and compare 1..4 (five events)
- The external events EXTEVNTA..J (10 events)
- All other timing units (e.g. timer B..F for timer A): compare 1, 2 and 4 (12 events)

Several events can be selected simultaneously to handle multiple reset sources. In this case, the multiple reset requests are ORed. When two counter reset events are generated within the same f_{HRTIM} clock cycle, the last counter reset is taken into account.

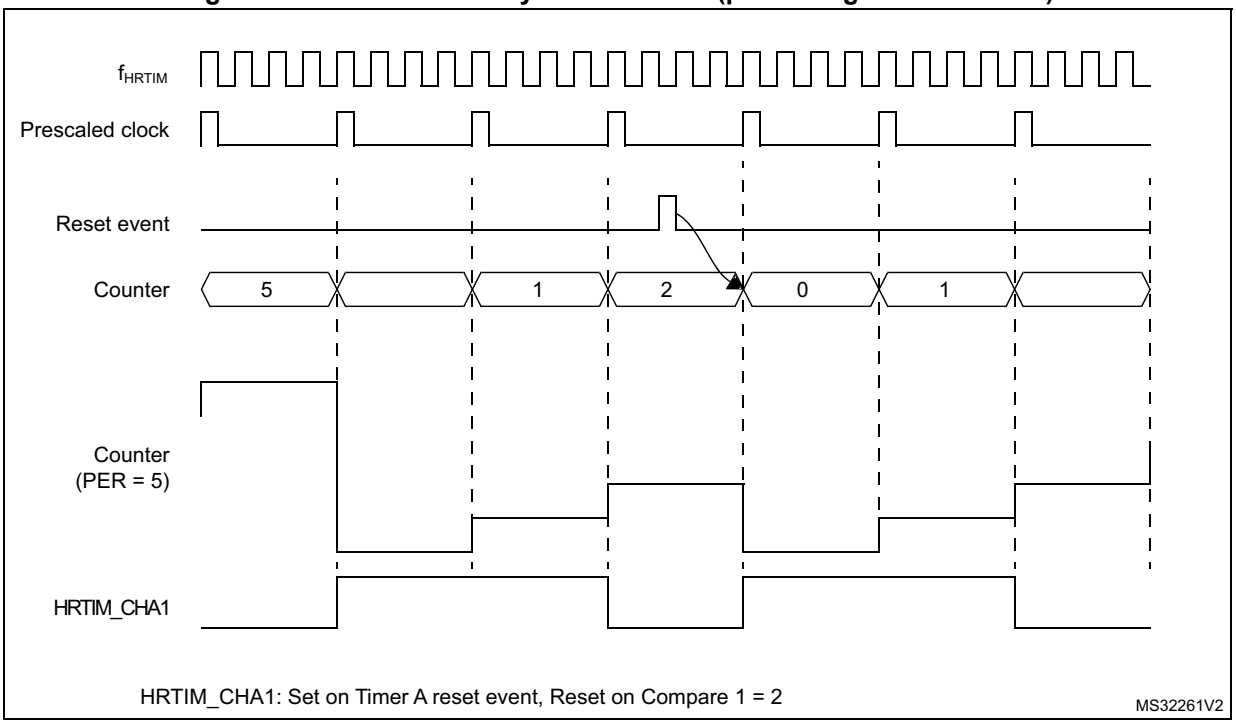
Additionally, it is possible to do a software reset of the counter using the TxRST bits in the HRTIM_CR2 register. These control bits are grouped into a single register to allow the simultaneous reset of several counters.

The reset requests are taken into account only once the related counters are enabled (TxCEN bit set).

When the f_{HRTIM} clock prescaling ratio is above 1, the counter reset event is delayed to the next active edge of the prescaled clock. This is used to maintain a jitterless waveform generation when an output transition is synchronized to the reset event (typically a constant Ton time converter).

Figure 426 shows how the reset is handled for a clock prescaling ratio of 4 (f_{HRTIM} divided by 4).

Figure 426. Timer reset resynchronization (prescaling ratio above 32)



Repetition counter

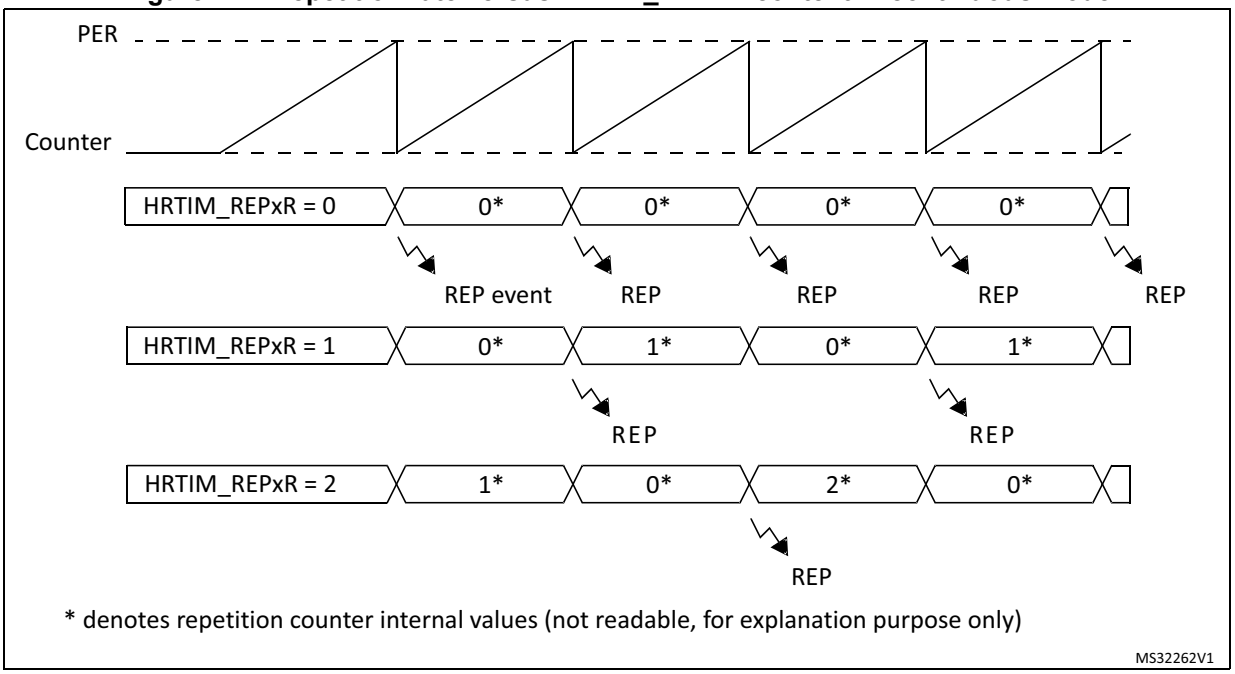
A common software practice is to have an interrupt generated when the period value is reached, so that the maximum amount of time is left for processing before the next period begins. The main purpose of the repetition counter is to adjust the period interrupt rate and off-load the CPU by decoupling the switching frequency and the interrupt frequency.

The timing units have a repetition counter. This counter cannot be read, but solely programmed with an autoreload value in the HRTIM_REPxR register.

The repetition counter is initialized with the content of the HRTIM_REPxR register when the timer is enabled (TXCEN bit set). Once the timer has been enabled, anytime the counter is cleared, either due to a reset event or due to a counter roll-over, the repetition counter is decreased. When it reaches zero, a REP interrupt or a DMA request is issued if enabled (REPIE and REPDE bits in the HRTIM_DIER register).

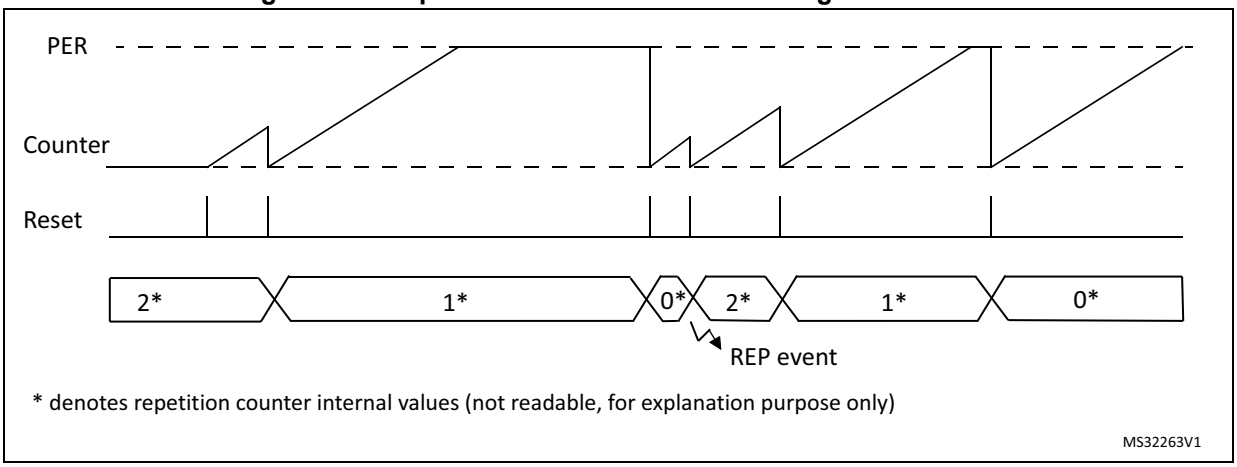
If the HRTIM_REPxR register is set to 0, an interrupt is generated for each and every period. For any value above 0, a REP interrupt is generated after (HRTIM_REPxR + 1) periods. *Figure 427* presents the repetition counter operation for various values, in continuous mode.

Figure 427. Repetition rate versus HRTIM_REPxR content in continuous mode



The repetition counter can also be used when the counter is reset before reaching the period value (variable frequency operation) either in continuous or in single-shot mode (*Figure 428* here-below). The reset causes the repetition counter to be decremented, at the exception of the very first start following counter enable (TxCEN bit set).

Figure 428. Repetition counter behavior in single-shot mode



A reset or start event from the HRTIM_SCIN[3:1] source causes the repetition to be decremented as any other reset. However, in SYNCIN-started single-shot mode (SYNCSTRTx bit set in the HRTIM_TIMxCR register), the repetition counter is decremented only on the first reset event following the period. Any subsequent restarted by a new request on HRTIM_SCIN[3:1] inputs.

Set/reset crossbar

A “set” event correspond to a transition to the output active state, while a “reset” event corresponds to a transition to the output inactive state.

The polarity of the waveform is defined in the output stage to accommodate positive or negative logic external components: an active level corresponds to a logic level 1 for a positive polarity (POLx = 0), and to a logic level 0 for a negative polarity (POLx = 1).

Each of the timing units handles the set/reset crossbar for two outputs. These two outputs can be set, reset, or toggled by up to 32 events that can be selected among the following sources:

- The timing unit: period, compare 1..4, register update (six events)
- The master timer: period, compare 1..4, HRTIM synchronization (six events)
- All other timing units (e.g. timer B..F for timer A): TIMEVNT1..9 (nine events described in *Table 339*)
- The external events EXTEVNTA..J (10 events)
- A software forcing (one event)

Note: In up/down mode (UDM bit set to 1), the counter period event is defined as per the OUTROM[1:0] bit setting.

The event sources are ORed and multiple events can be simultaneously selected.

Each output is controlled by two 32-bit registers, one coding for the set (HRTIM_SETxyR) and another one for the reset (HRTIM_RSTxyR), where x stands for the timing unit: A..F and y stands for the output 1or 2 (e.g. HRTIM_SETA1R, HRTIM_RSTC2R,...).

If the same event is selected for both set and reset, it toggles the output. It is not possible to toggle the output state more than one time per t_{HRTIM} period: in case of two consecutive toggling events within the same cycle, only the first one is considered.

The set and reset requests are taken into account only once the counter is enabled (TxCEN bit set), except if the software is forcing a request to allow the prepositioning of the outputs at timer startup.

Table 339 summarizes the events from other timing units that can be used to set and reset the outputs. The number corresponds to the timer events (such as TIMEVNTx) listed in the register, and empty locations are indicating nonavailable events.

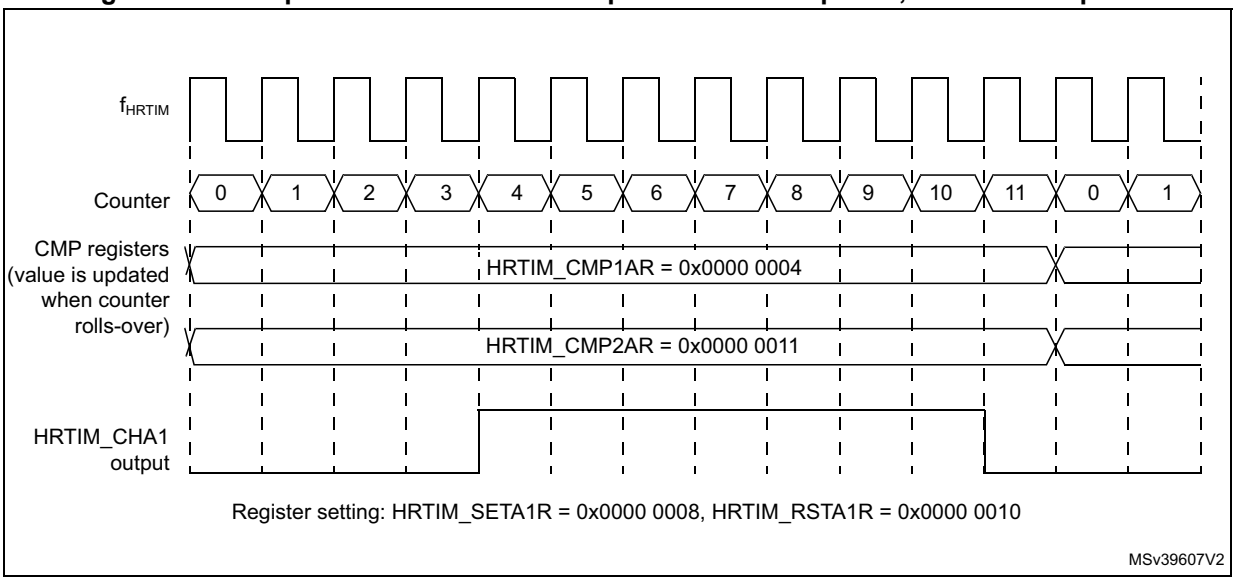
For instance, timer A outputs can be set or reset by the following events: timer B compare 1 and 2, timer C compare 2 and 3,... and timer E compare 3 is listed as TIMEVNT7 in HRTIM_SETA1R.

Table 339. Events mapping across timer A to F

Source		Timer A				Timer B				Timer C				Timer D				Timer E				Timer F			
		CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4
Destination	TA	-	-	-	-	1	2	-	-	-	3	4	-	5	6	-	-	-	-	7	8	-	-	-	9
	TB	1	2	-	-	-	-	-	-	-	-	3	4	-	-	5	6	7	8	-	-	-	-	9	-
	TC	-	1	2	-	-	3	4	-	-	-	-	-	-	5	-	6	-	-	7	8	-	9	-	-
	TD	1	-	-	2	-	3	-	4	-	-	-	5	-	-	-	-	6	-	-	7	8	-	9	-
	TE	-	-	-	1	-	-	2	3	4	5	-	-	6	7	-	-	-	-	-	-	-	-	8	9
	TF	-	-	1	-	2	-	-	3	4	-	-	5	-	-	6	7	-	8	9	-	-	-	-	-

Figure 429 represents how a PWM signal is generated using two compare events.

Figure 429. Compare events action on outputs: set on compare 1, reset on compare 2



Set/reset on update events

Half mode

This mode aims at generating a square signal with a fixed 50% duty cycle and variable frequency (typically for converters using resonant topologies). It is used to have the duty cycle automatically forced to half of the period value when a new period is programmed.

This mode is enabled by writing HALF bit to 1 in the HRTIM_TIMxCR register. When the HRTIM_PERxR register is written, it causes an automatic update of the compare one value with HRTIM_PERxR/2 value.

The output on which a square wave is generated must be programmed to have one transition on the CMP1 event, and one transition on the period event, as follows:

- HRTIM_SETxyR = 0x0000 0008, HRTIM_RSTxyR = 0x0000 0004, or
- HRTIM_SETxyR = 0x0000 0004, HRTIM_RSTxyR = 0x0000 0008

The HALF mode overrides the content of the HRTIM_CMP1xR register. The access to the HRTIM_PERxR register only causes compare 1 internal register to be updated. The user-accessible HRTIM_CMP1xR register is not updated with the HRTIM_PERxR / 2 value.

When the preload is enabled (PREEN = 1), compare 1 active register is refreshed on the update event. If the preload is disabled (PREEN= 0), compare 1 active register is updated as soon as HRTIM_PERxR is written.

The period must be greater than or equal to six periods of the f_{HRTIM} clock when the HALF mode is enabled.

Interleaved mode

This mode complements the Half mode and helps the implementation of interleaved topologies.

It is used to recompute automatically the content of compare registers when the HRTIM_PERxR value is updated.

The selection is done using the HALF bit and the INTLVD[1:0] bits in HRTIM_MCR and HRTIM_TIMxCR registers, as shown on the *Table 340* below.

Table 340. Interleaved mode selection

HALF bit	INTLVD [1:0] bits	Mode
0	00	Disabled
0	01	Triple interleaved (120°)
0	10	Quad interleaved (90°)
0	11	Reserved
1	xx	Dual interleaved (180°)

Table 341 gives the compare values for the three available modes. The content of the compare registers is overridden. The corresponding compare events can be used to trigger an output set/reset or to reset a slave timer.

Table 341. Compare 1..3 values in interleaved mode

Mode	Dual interleaved 180°	Triple interleaved 120°	Quad interleaved 90°
CMP1 value	PERxR/2	PERxR/3	PERxR/4
CMP2 value	Not affected	2x (PERxR/3)	PERxR/2
CMP3 value	Not affected	Not affected	3x (PERxR/4)

Note: In half and interleaved modes, the compare registers are controlled by hardware and writing them has no effect. However the written value is stored in the preload register and becomes active on the update event following the exit of these modes.

Note: The triple and quad interleaved modes must not be used simultaneously with other modes using CMP2 (dual channel dac trigger and triggered-half modes).

Null duty cycle exception case

The high-resolution behavior is not supported for pulses narrower than three t_{HRTIM} periods (see *Section 33.3.7: Set/reset events priorities and narrow pulses management*) and any value strictly below 3 periods of the f_{HRTIM} clock in the HRTIM_TIMxCMPy register is forbidden (see *33.5.19: HRTIM timer x compare 1 register (HRTIM_CMP1xR) (x = A to F)*).

However, it is possible to skip an output pulse and have a null duty cycle by simply writing a null value in the HRTIM_TIMxCMPy registers, if and only if the following conditions are met:

- The output SET event is generated by the PERIOD event
- The output RESET if generated by the compare event
- The compare event is active within the timer unit itself, and not used for other timing units

For any other use case, this can be done by programming the SET and RESET events with the very same compare values, above three periods of the f_{HRTIM} clock. In this case, the output is reset forced (following the set/reset priority scheme defined in the *Section 33.3.7: Set/reset events priorities and narrow pulses management*).

When a null value is programmed, the CMPx flags are set, and the related interrupt is issued if enabled. An event is also generated on synchronization output when SYNCSRC= 11 (synchronization generated on TIM1 CMP1 event).

The following limitations apply when a null value is programmed:

- ADC and burst mode triggers are not generated
- The CMP1 cannot be used for blanking events or faults

Swap mode

This mode is used to swap the two outputs with a single bit access: the output 1 signal is connected to the output 2 pin and the output 2 signal is connected to output 1 pin. The output swap is triggered with the SWPx bits in the HRTIM_CR2 register and is effective on the next update event.

The outputs are swapped prior to the set/reset crossbar unit, as follows:

- If SWPx = 0, HRTIM_SETx1R and HRTIM_RSTx1R are coding for the output 1, HRTIM_SETx2R and HRTIM_RSTx2R are coding for the output 2
- If SWPx = 1, HRTIM_SETx1R and HRTIM_RSTx1R are coding for the output 2, HRTIM_SETx2R and HRTIM_RSTx2R are coding for the output 1

The swap mode is only affecting the preload register, and not the active registers.

Note: The preload mode must be enabled when using the swap mode.

Consequently, it does not modify the auxiliary outputs in parallel with the regular outputs going to the output stage (see *Section 33.3.18* for details). They provide the following internal status, events, and signals:

- O1CPY, O2CPY, SETxy and RSTxy status flags, together with the corresponding interrupts and DMA requests
- Capture triggers upon output set/reset (TA2, TB2, TC2, TD2, TE2, TF2)
- External event filters generated with a Tx2 output copy

For instance the SETx1 flag is related to the output 1 when SWP = 0 and is related to the output 2 when SWPx = 1.

Similarly, the swap mode does not change the attribution of control bits in the HRTIM_OUTxR register (DIDLx, CHPx, FAULTx[1:0], IDLESx, POLx bits). For instance, the POL1 bit controls the output 1 polarity whatever the SWP bit value.

Note: The SWPx bits are ignored in push-pull mode (PSHPLL = 1 in the HRTIM_TIMxCR register).

Capture

The timing unit has the capability to capture the counter value, triggered by internal and external events. The purpose is to:

- Measure events arrival timings or occurrence intervals
- Update compare 2 and compare 4 values in autodelayed mode (see *Section : Autodelayed mode*).

The capture is done with f_{HRTIM} resolution.

The timer has two capture registers: HRTIM_CPT1xR and HRTIM_CPT2xR. The capture triggers are programmed in the HRTIM_CPT1xCR and HRTIM_CPT2xCR registers.

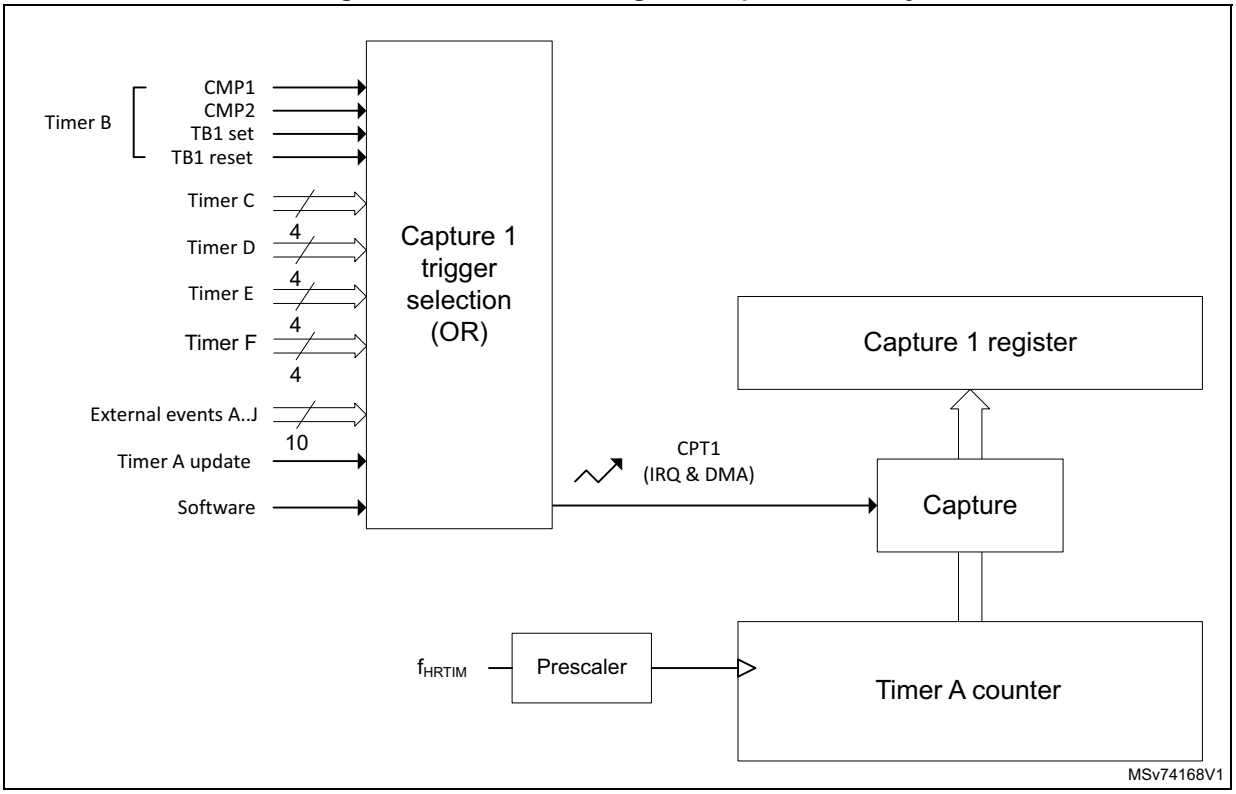
The capture of the timing unit counter can be triggered by up to 28 events that can be selected simultaneously in the HRTIM_CPT1xCR and HRTIM_CPT2xCR registers, among the following sources:

- The external events, EXTEVNTA..J (10 events)
- All other timing units (e.g. timer B..F for timer A): compare 1, 2 and output 1 set/reset events (16 events)
- The timing unit: update (one event)
- A software capture (one event)

Several events can be selected simultaneously to handle multiple capture triggers. In this case, the concurrent trigger requests are ORed. The capture can generate an interrupt or a DMA request when CPTxIE and CPTxDE bits are set in the HRTIM_TIMxDIER register.

Overcapture is not prevented by the circuitry: a new capture is triggered even if the previous value was not read, or if the capture flag was not cleared.

Figure 430. Timer A timing unit capture circuitry



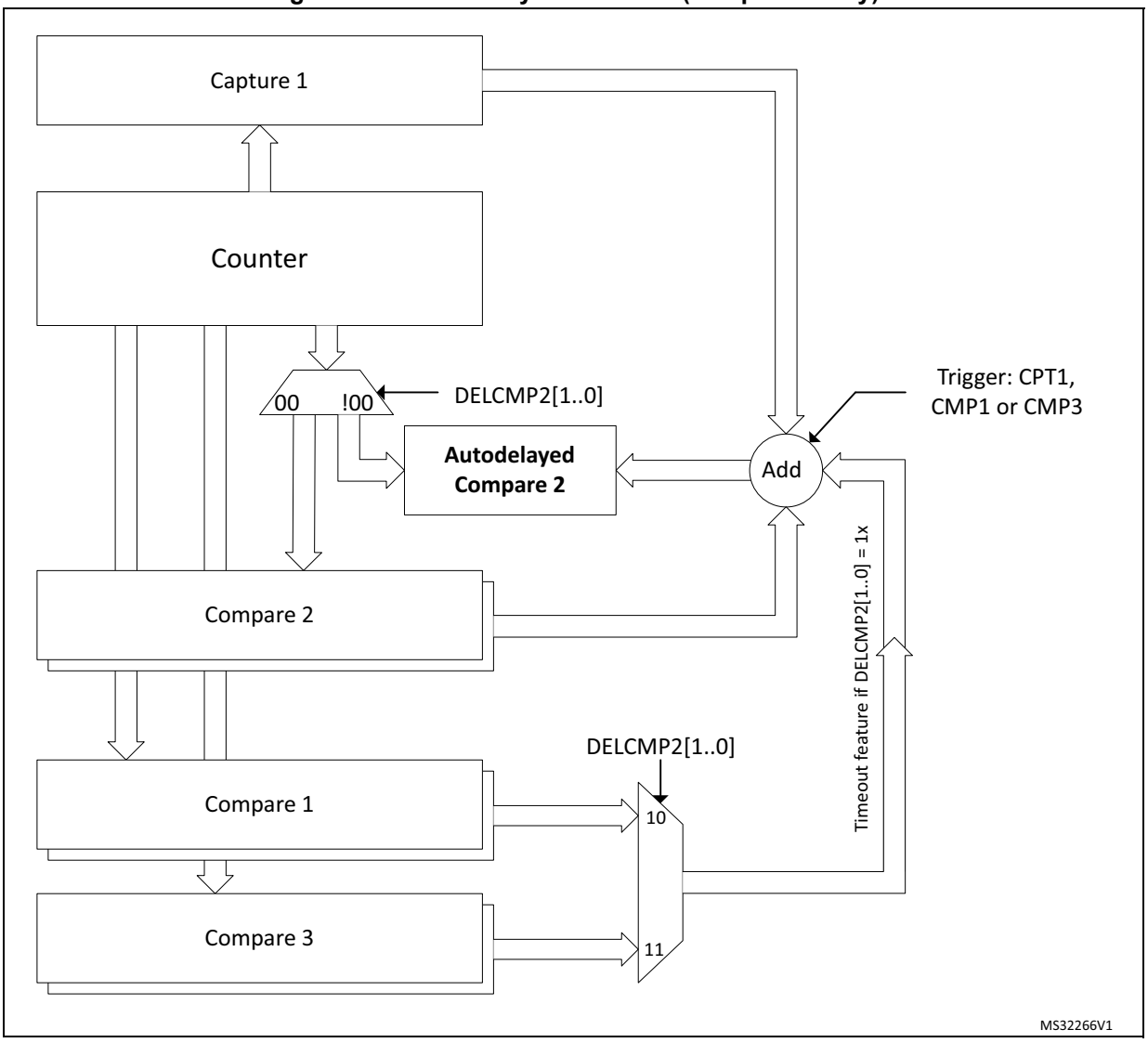
Autodelayed mode

This mode is used to have compare events generated relatively to capture events, so that for instance an output change can happen with a programmed timing following a capture. In this case, the compare match occurs independently from the timer counter value. It enables the generation of waveforms with timings synchronized to external events without the need of software computation and interrupt servicing.

As long as no capture is triggered, the content of the HRTIM_CMPxR register is ignored (no compare event is generated when the counter value matches the compare value. Once the capture is triggered, the compare value programmed in HRTIM_CMPxR is summed with the captured counter value in HRTIM_CPTxyR, and it updates the internal autodelayed compare register, as seen on *Figure 431*. The autodelayed compare register is internal to the timing unit and cannot be read. The HRTIM_CMPxR preload register is not modified after the calculation.

This feature is available only for compare 2 and compare 4 registers. Compare 2 is associated with capture 1, while compare 4 is associated with capture 2. HRTIM_CMP2xR and HRTIM_CMP4xR compares cannot be programmed with a value below three f_{HRTIM} clock periods, as in the regular mode.

Figure 431. Auto-delayed overview (compare 2 only)



The autodelayed compare is only valid from the capture up to the period event: once the counter has reached the period value, the system is rearmed with compare disabled until a capture occurs.

DELCMP2[1:0] and DELCMP4[1:0] bits in HRTIM_TIMxCR register are used to configure the autodelayed mode as follows:

- 00
  Regular compare mode: HRTIM_CMP2xR and HRTIM_CMP4xR register contents are directly compared with the counter value.
- 01
  Autodelayed mode: compare 2 and compare 4 values are recomputed and used for comparison with the counter after a capture 1/2 event.
- 10 or 11
  Autodelayed mode with timeout: compare 2 and compare 4 values are recomputed and used for comparison with the counter after a capture 1/2 event or after a compare 1

match (DELCMPx[1:0]= 10) or a compare 3 match (DELCMPx[1:0]= 11) to have a timeout function if capture 1/2 event is missing.

When the capture occurs, the comparison is done with the (HRTIM_CMP2/4xR + HRTIM_CPT1/2xR) value. If no capture is triggered within the period, the behavior depends on the DELCMPx[1:0] value:

- DELCMPx[1:0] = 01: the compare event is not generated
- DELCMPx[1:0] = 10 or 11: the comparison is done with the sum of the two compares (for instance HRTIM_CMP2xR + HRTIM_CMP1xR). The captures are not taken into account if they are triggered after CMPx + CMP1 (resp. CMPx + CMP3).

The captures are enabled again at the beginning of the next PWM period.

If the result of the autodelayed summation is above 0xFFFF (overflow), the value is ignored and no compare event is generated until a new period is started.

Note: DELCMPx[1:0] bitfield must be reset when reprogrammed from one value to the other to reinitialize properly the autodelayed mechanism, for instance:

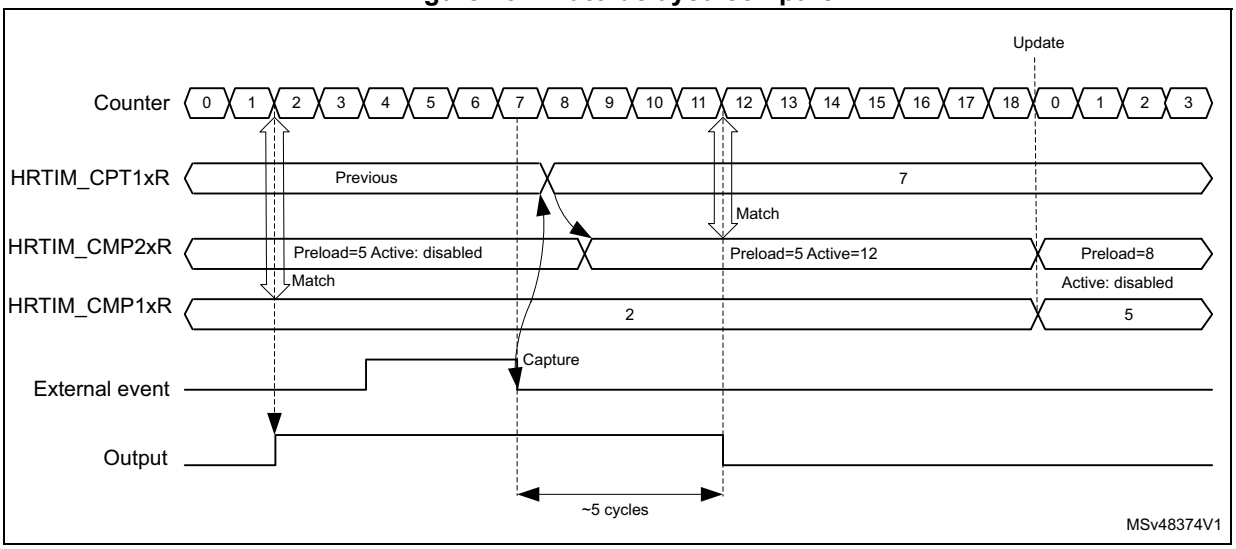
- DELCMPx[1:0] = 10
- DELCMPx[1:0] = 00
- DELCMPx[1:0] = 11

As an example, *Figure 432* shows how the following signal is generated:

- Output set when the counter is equal to compare 1 value
- Output reset 5 cycles after a falling edge on a given external event

Note: To simplify the figure, the external event signal is shown without any resynchronization delay: practically, there is a delay of one to two f_{HRTIM} clock periods between the falling edge and the capture event due to an internal resynchronization stage which is necessary to process external input signals.

Figure 432. Auto-delayed compare



A regular compare channel (e.g. compare 1) is used for the output set: as soon as the counter matches the content of the compare register, the output goes to its active state.

A delayed compare is used for the output reset: the compare event can be generated only if a capture event has occurred. No event is generated when the counter matches the delayed

compare value (counter = 4). Once the capture event has been triggered by the external event, the content of the capture register is summed to the delayed compare value to have the new compare value. In the example, the autodelayed value 4 is summed to the capture equal to 7 to give a value of 12 in the autodelayed compare register. From this time on, the compare event can be generated and happens when the counter is equal to 12, causing the output to be reset.

Overcapture management in autodelayed mode

Overcapture is prevented when the autodelayed mode is enabled (DELCMPx[1:0] = 01, 10, 11).

When multiple capture requests occur within the same counting period, only the first capture is taken into account to compute the autodelayed compare value. A new capture is possible only:

- Once the autodelayed compare has matched the counter value (compare event)
- Once the counter has rolled over (period)
- Once the timer has been reset

Changing autodelayed compare values

When the autodelayed compare value is preloaded (PREEN bit set), the new compare value is taken into account on the next coming update event (for instance on the period event), regardless of when the compare register was written and if the capture occurred (see *Figure 432*, where the delay is changed when the counter rolls over).

When the preload is disabled (PREEN bit reset), the new compare value is taken into account immediately, even if it is modified after the capture event has occurred, as per the example below:

1. At t1, DELCMP2 = 1.
2. At t2, CMP2_act = 0x40 => comparison disabled
3. At t3, a capture event occurs capturing the value CPTR1 = 0x20. => comparison enabled, compare value = 0x60
4. At t4, CMP2_act = 0x100 (before the counter reached value CPTR1 + 0x40) => comparison still enabled, new compare value = 0x120
5. At t5, the counter reaches the period value => comparison disabled, cmp2_act = 0x100

Similarly, if the CMP1(CMP3) value changes while DELCMPx = 10 or 11, and preload is disabled:

1. At t1, DELCMP2 = 2.
2. At t2, CMP2_act = 0x40 => comparison disabled
3. At t3, CMP3 event occurs - CMP3_act = 0x50 before capture 1 event occurs => comparison enabled, compare value = 0x90
4. At t4, CMP3_act = 0x100 (before the counter reached value 0x90) => comparison still enabled, compare 2 event occurs at = 0x140

Triggered-half mode

The purpose of this mode is to allow the synchronization of two interleaved converters that have variable frequency operation, and require a 180° phase-shift. The basic principle is to have a master-slave system. The slave converter synchronization is continuously adjusted based on the previous switching period of the master converter.

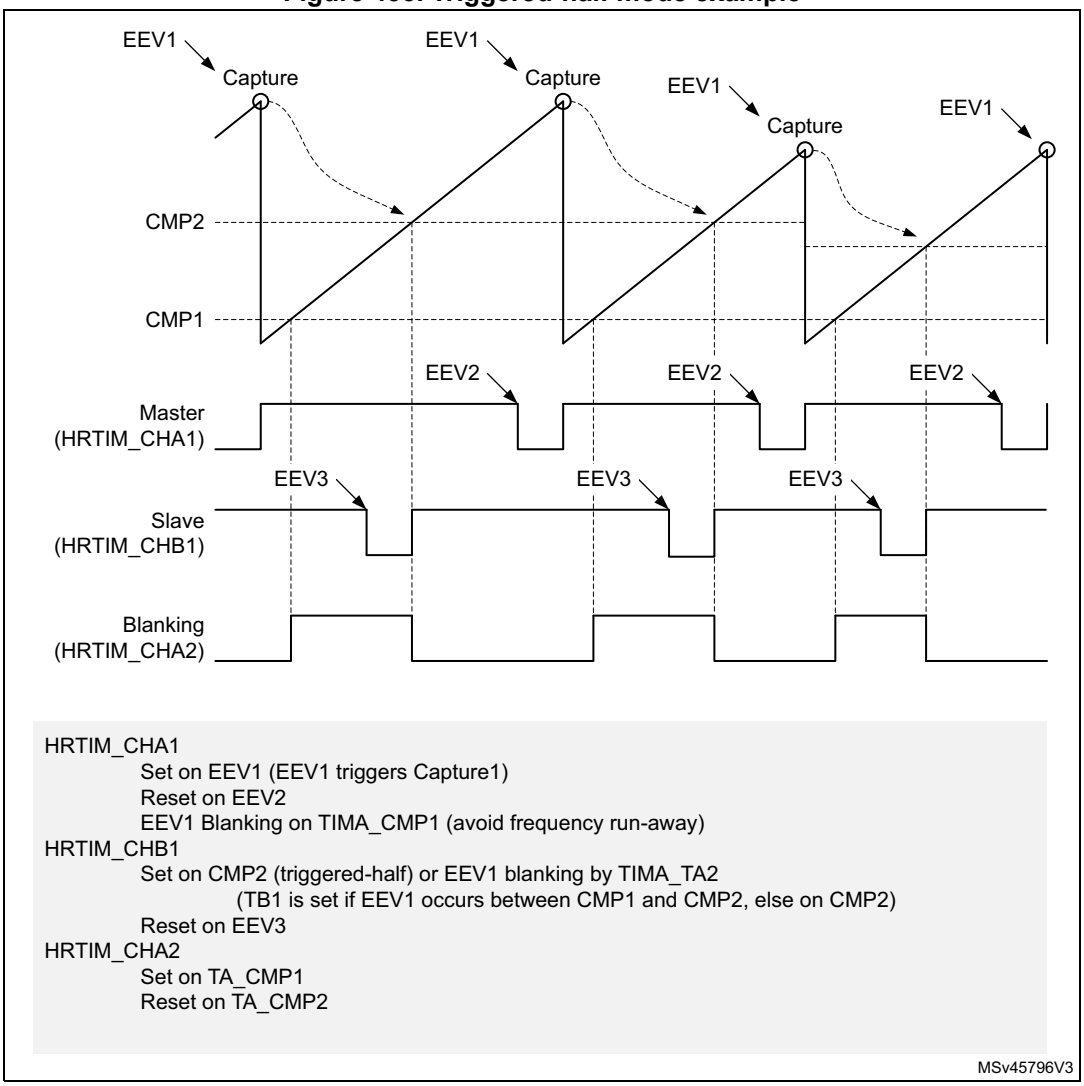
This is done using the capture unit. The switching period of the master converter is captured, divided by two and then stored in the compare 2 register by hardware. The compare 2 register contains a value equal to half of the captured period, which is the master converter switching period. The compare 2 event can then be used to trigger a second timing unit that manages the slave converter.

This mode is enabled by setting the TRGHLF bit in the HRTIM_TIMxCR2 register. This bit cannot be changed once the timer is operating (TxEN bit set).

The triggered-half mode must not be used simultaneously with other modes using CMP2 (dual channel DAC trigger, interleaved and balanced idle modes).

The initial value CMP2 can be written by the user, but is ignored once the first capture is triggered. The preload mechanism is disabled for CMP2 when the TRGHLF bit is reset.

Figure 433. Triggered-half mode example



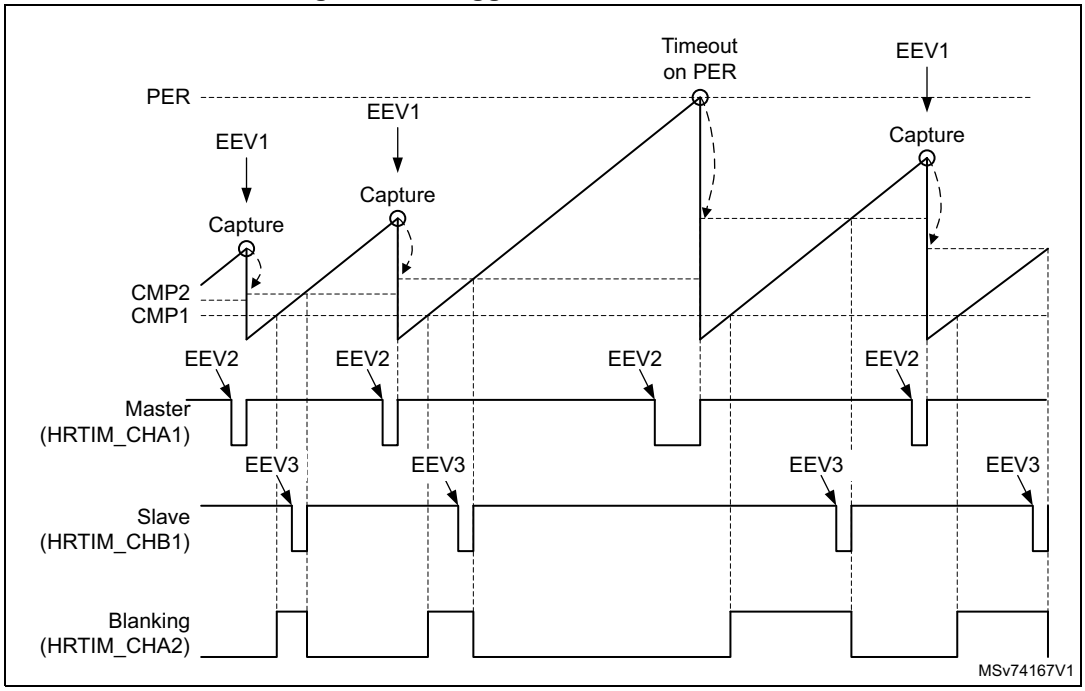
Note: In triggered half mode, the compare2 register is controlled by hardware and writing it has no effect. However the written value is stored in the preload register and becomes active on the update event following the exit of this mode.

It is possible to generate a timeout event when the external event is missing, as shown on *Figure 434* below. This mode is enabled with the TRGHLFT in HRTIM_CR2 register.

No capture event is generated when a timeout occurs. A timeout situation can be detected using the PER interrupt.

The timeout occurs regardless of the repetition value.

Figure 434. Triggered-half mode timeout



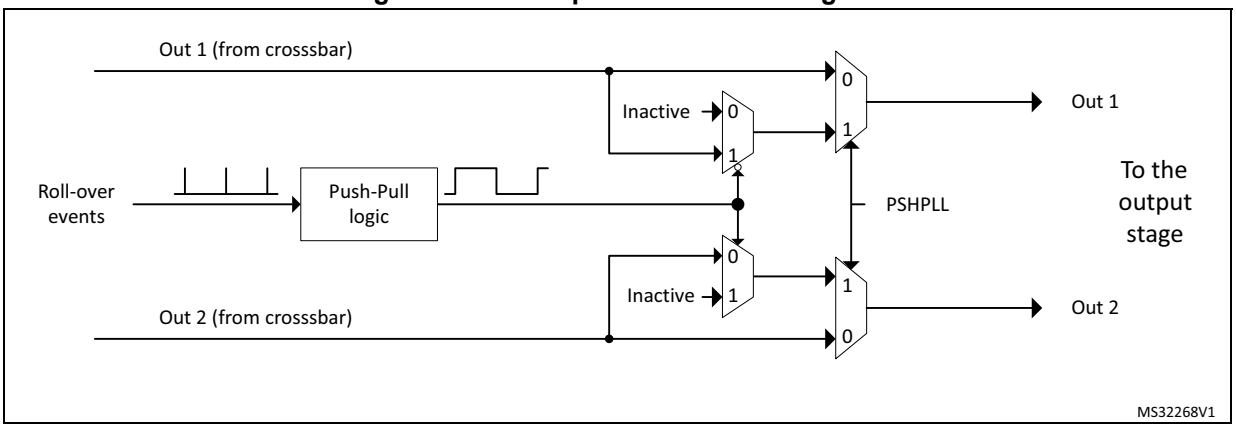
Push-pull mode

This mode primarily aims at driving converters using push-pull topologies. It also needs to be enabled when the delayed idle protection is required, typically for resonant converters (refer to *Section 33.3.10: Delayed protection*).

The push-pull mode is enabled by setting the PSHPLL bit in the HRTIM_TIMxCR register.

It applies the signals generated by the crossbar to output 1 and output 2 alternatively, on the period basis, maintaining the other output to its inactive state. The redirection rate (push-pull frequency) is defined by the timer's period event, as shown on *Figure 435*. The push-pull period is twice the timer counting period.

Figure 435. Push-pull mode block diagram



The push-pull mode is available when the timer operates in continuous mode and in single-shot mode. It is necessary to disable the timer to stop a push-pull operation and to reset the counter before reenabling it.

To get a correct behavior, the event selected as source of the counter reset must be also selected for setting (or resetting) the output. It must set the output if the output is set on period, else it must reset it. If it is not done, the output switching from its inactive period to its active period may be incorrect (may unexpectedly rise or may unexpectedly stay low).

The signal shape is defined using HRTIM_SETxyR and HRTIM_RSTxyR for both outputs. It is necessary to have HRTIM_SETx1R = HRTIM_SETx2R and HRTIM_RSTx1R = HRTIM_RSTx2R to have both outputs with identical waveforms and to achieve a balanced operation. Still, it is possible to have different programming on both outputs for other uses.

The CPPSAT status bit in HRTIM_TIMxISR indicates on which output the signal is currently active. CPPSTAT is reset when the push-pull mode is disabled.

In the example given on *Figure 436*, the timer internal waveform is defined as follows:

- Output set on period event
- Output reset on compare 1 match event

Figure 436. Push-pull mode example

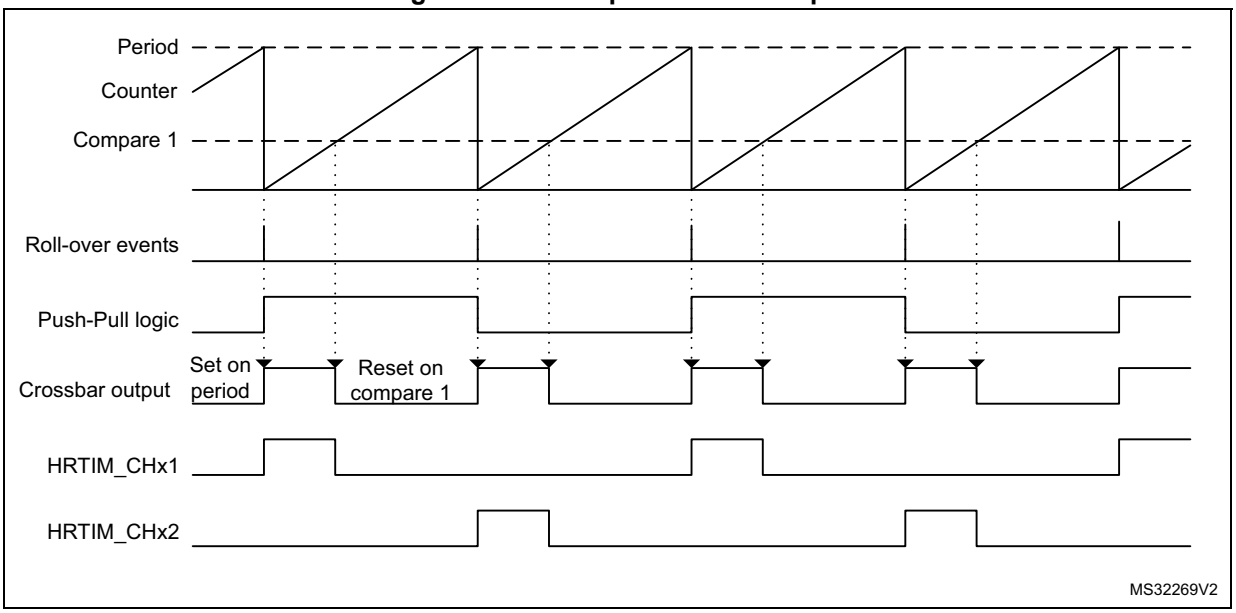
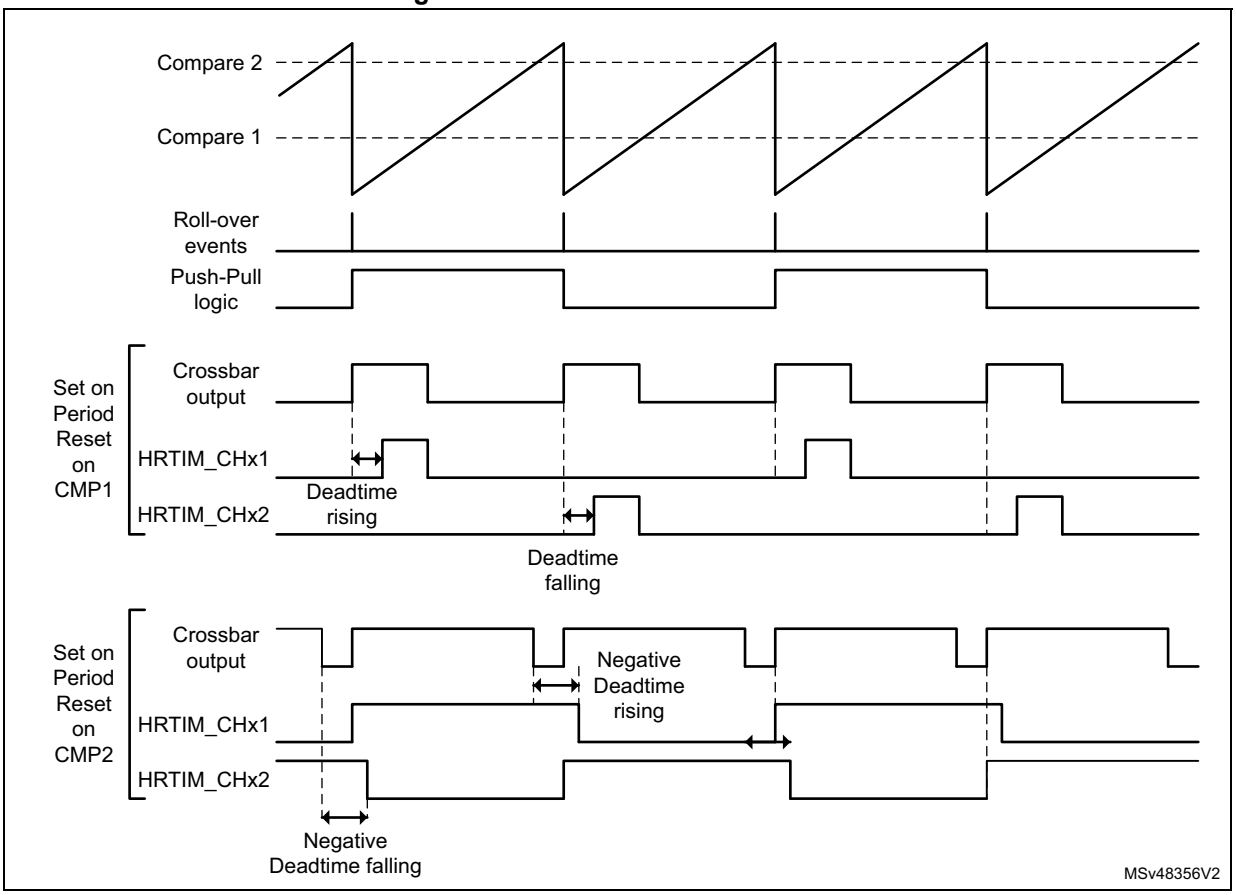


Figure 437 shows how the deadtime is inserted in push-pull mode, for both positive and negative deadtimes. In this case, the outputs are not complemented anymore, and the deadtime is applied on each output individually (both ouput 1 and output 2 of the crossbar are used).

Figure 437. Push-Pull with deadtime

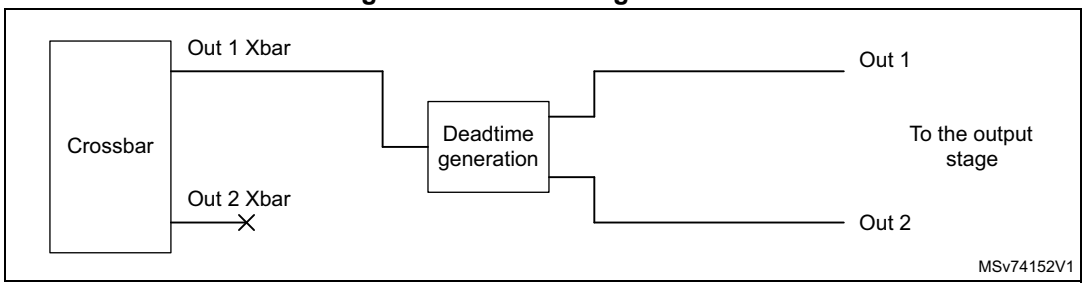


Deadtime

A deadtime unit is used to generate a couple of complementary signals from a single reference waveform, with programmable delays between active state transitions. This is commonly used for topologies using half-bridges or full bridges. It simplifies the software: only one waveform is programmed and controlled to drive two outputs.

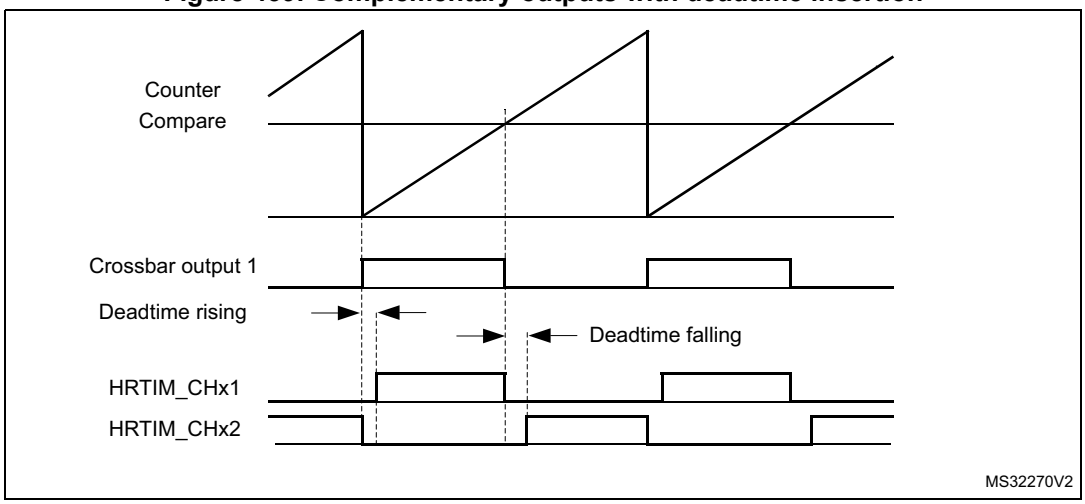
The deadtime is enabled by setting DTEN bit in HRTIM_OUTxR register. The complementary signals are built based on the reference waveform defined for output 1, using HRTIM_SETx1R and HRTIM_RSTx1R registers: HRTIM_SETx2R and HRTIM_RSTx2R registers are not significant when DTEN bit is set, as shown on *Figure 438*.

Figure 438. Deadtime generation



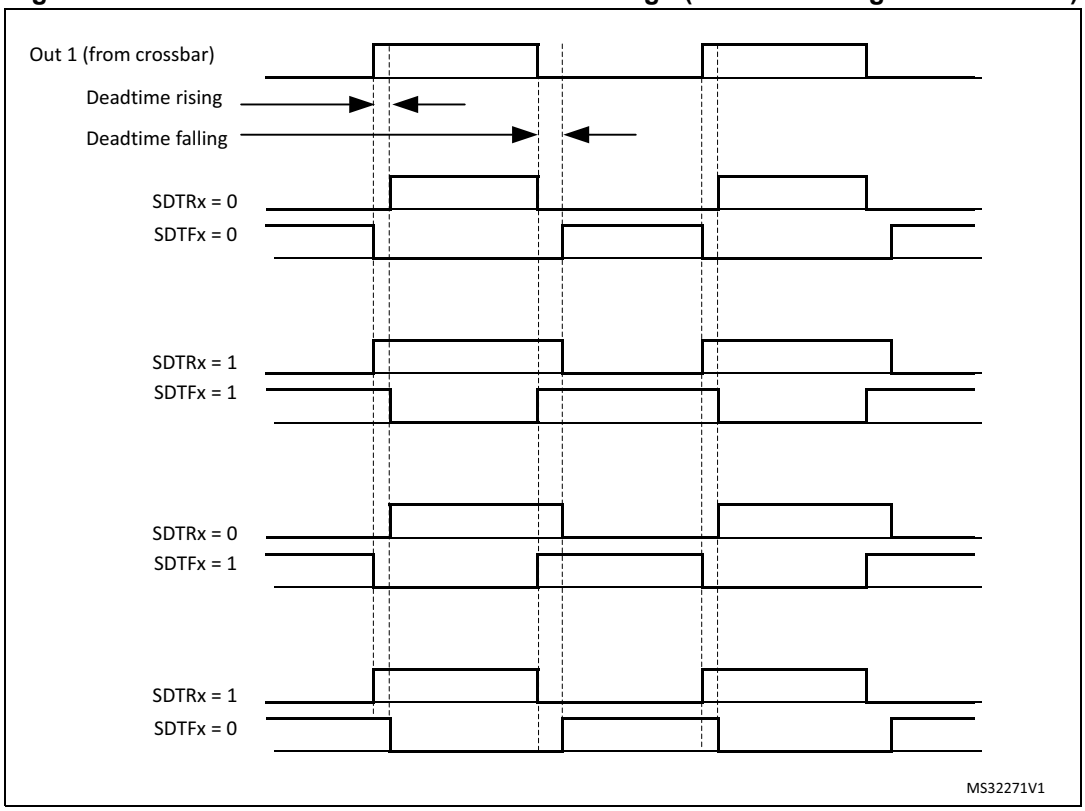
Two deadtimes can be defined in relationship with the rising edge and the falling edge of the reference waveform, as in *Figure 439*.

Figure 439. Complementary outputs with deadtime insertion



Negative deadtime values can be defined when some control overlap is required. This is done using the deadtime sign bits (SDTFx and SDTRx bits in HRTIM_DTxR register). *Figure 440* shows complementary signal waveforms depending on respective signs.

Figure 440. Deadtime insertion versus deadtime sign (1 indicates negative deadtime)



The deadtime values are defined with DTFx[8:0] and DTRx[8:0] bitfields and based on a specific clock prescaled according to DTPRSC[2:0] bits, as follows:

$t_{DTx} = \pm DTx[8:0] \times t_{DTG}$

where x is either R or F and $t_{DTG} = (2^{(DTPRSC[2:0])}) \times t_{HRTIM}$.

Table 342 gives the resolution and maximum absolute values depending on the prescaler value.

Table 342. Deadtime resolution and max absolute values

DTPRSC[2:0]	t_{DTG}	t_{DTx} max	f_{HRTIM}= 520 MHz	
			t_{DTG} (ns)	$\|t_{DTx}\|$ max (µs)
011	t_{HRTIM}	511 * t_{DTG}	1.9	0.98
100	2 * t_{HRTIM}		3.8	1.96
101	4 * t_{HRTIM}		7.7	3.93
110	8 * t_{HRTIM}		15.4	7.86
111	16 * t_{HRTIM}		30.8	15.7

Figure 441 to *Figure 444* present how the deadtime generator behaves for reference waveforms with pulsewidth below the deadtime values, for all deadtime configurations.

Figure 441. Complementary outputs for low pulsewidth (SDTRx = SDTFx = 0)

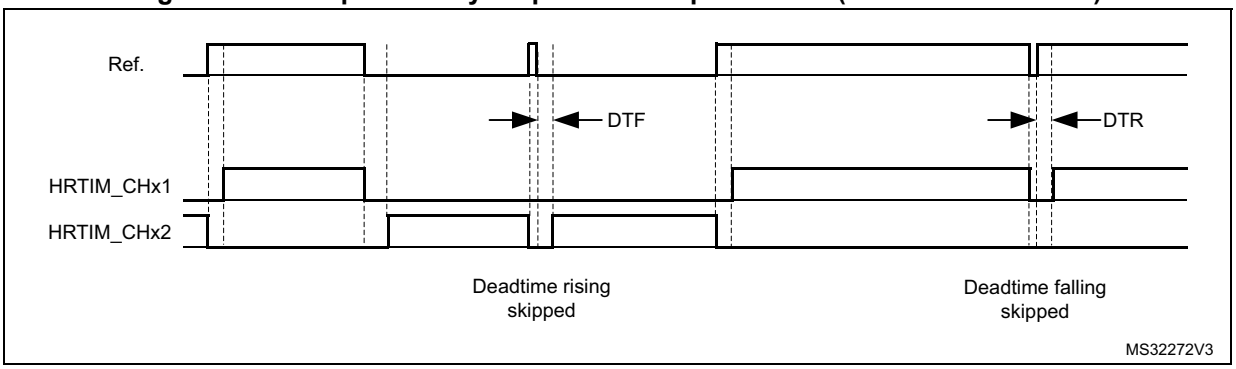


Figure 442. Complementary outputs for low pulsewidth (SDTRx = SDTFx = 1)

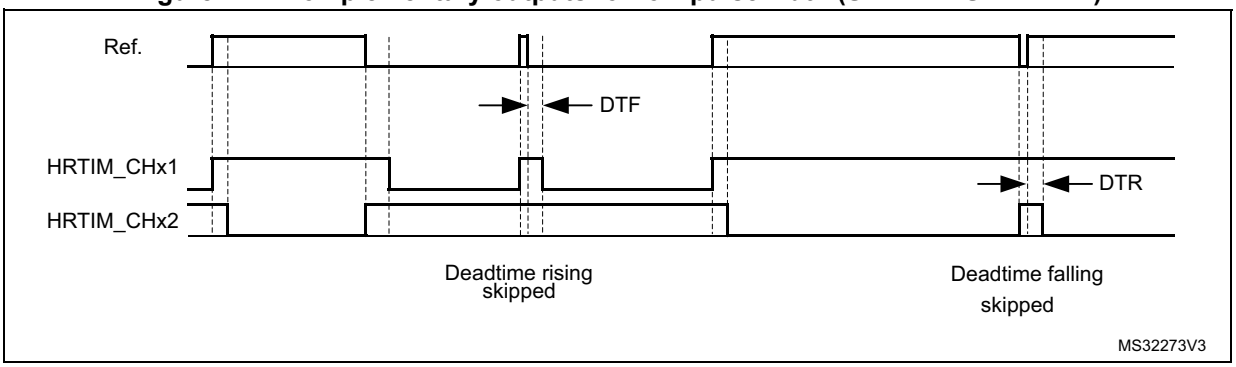


Figure 443. Complementary outputs for low pulsewidth (SDTRx = 0, SDTFx = 1)

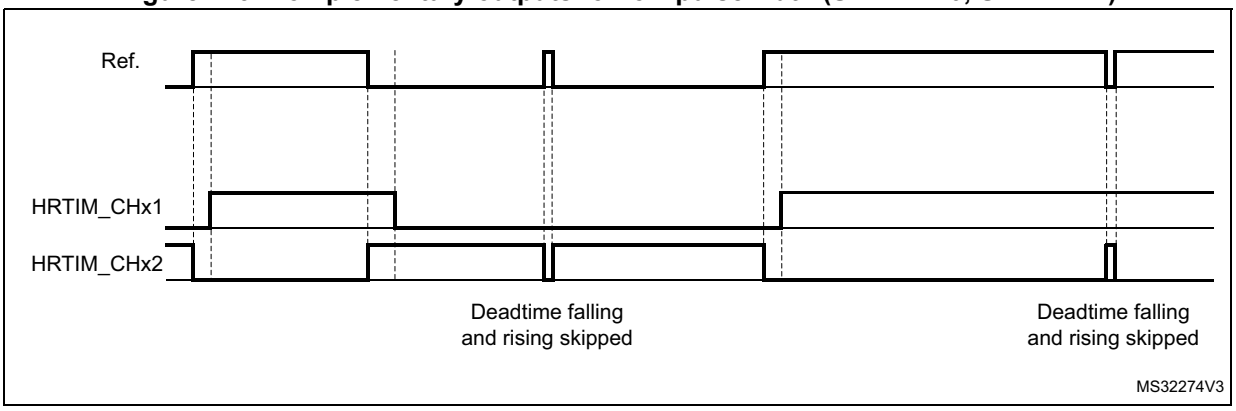
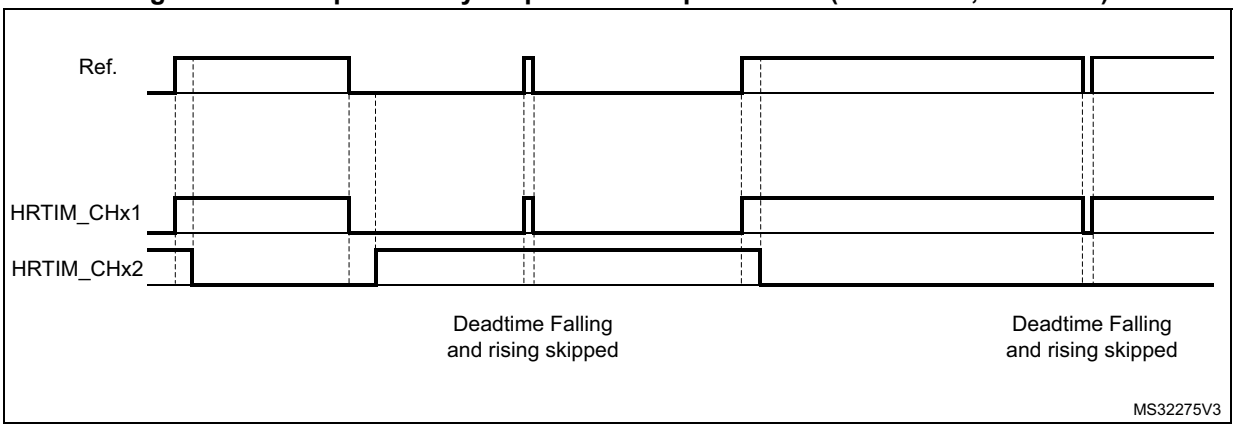


Figure 444. Complementary outputs for low pulsewidth (SDTRx = 1, SDTFx=0)



For safety purposes, it is possible to prevent any spurious write into the deadtime registers by locking the sign and/or the value of the deadtime using DTFLKx, DTRLKx, DTFSLKx, and DTRSLKx. Once these bits are set, the related bits and bitfields are becoming read only until the next system reset.

The DTEN bit is preloaded, and can be written while the timer is enabled.

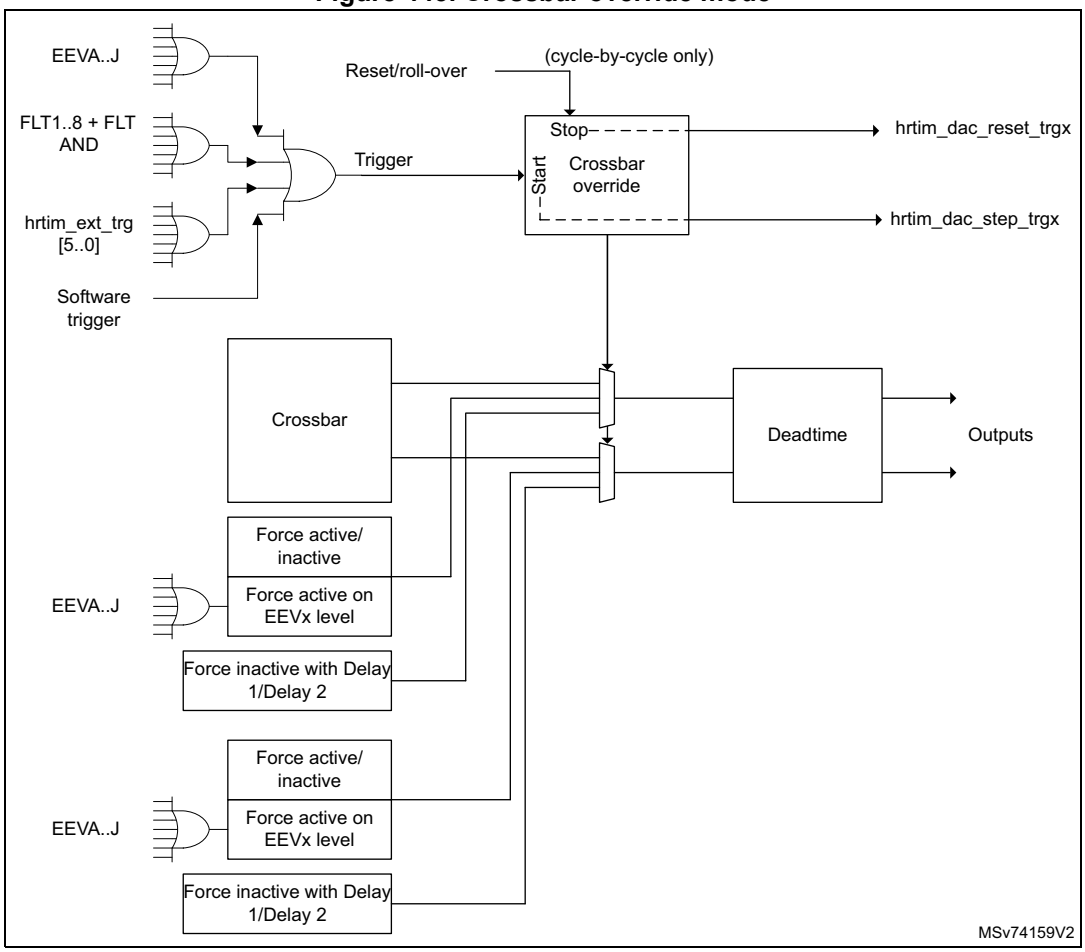
Caution: DTEN bit must not be changed when the timer outputs are set/reset by another timer (while TxEN is reset)

Note: For the particular case where DTEN must be set while the burst mode is enabled with a deadtime upon entry (BME = 1, DIDL = 1, IDLEM = 1), it is necessary to force the two outputs in their IDLES state by software commands (SST, RST bits) before setting DTEN bit. This is to avoid any side effect resulting from a burst mode entry that would happen immediately before a deadtime enable.

Crossbar override

This mode gives the possibility to override the output crossbar programming and take over the output control to force a level based on a fixed setting or based on a given external event or fault, as shown on *Figure 445* below.

Figure 445. Crossbar override mode



This mode is triggered on one or multiple sources selected in HRTIM_COTRGxR register:

- Software trigger
- Fault 1..8 and AND combined faults inputs
- External events A..J
- Six external crossbar override triggers hrtim_ext_trg[5:0]

Once the crossbar override mode is entered, specific external events will force high or low the outputs when active.

The crossbar override logic provides 2 options:

- Latched: once triggered, the crossbar override mode is maintained until a software intervention to disengage it.
- Cycle-by-cycle mode: the crossbar override mode exits automatically and normal operation is resumed by hardware. Typically, the system will re-arm once the over-

current condition has disappeared.
This mode is only applicable in force on event mode (COFMx[2:0] = 100).

In cycle-by-cycle mode (COM bit reset in HRTIM_TIMxCR2), the Crossbar override function is disengaged when both:

- The trigger event is inactive
- The next reset/roll-over happens (conditioned by ROM[1:0] in center-aligned mode)

In latched mode (COM bit set in HRTIM_TIMxCR2), the crossbar override mode can be disengaged in two ways:

- With the rearm bit COR in HRTIM_TIMxCR2. The rearming will be effective only of the trigerring condition has disappeared (status can be polled using the COTS bit in the HRTIM_TIMxISR register).
- With the COE bit reset, to disable it permanently.

The DAC values can be updated synchronously with the crossbar override mode triggering. The triggers are generated when the DCDCOE (Dual Channel DAC Crossbar Override Enable) bit is set in HRTIM_TIMxCR2 (the DCDE value is overridden when DCDCOE=1).

- hrtim_dac_reset_trgx triggers are generated only in cycle-by-cycle mode when the crossbar override stops, and DCDR setting applies.
- hrtim_dac_step_trgx triggers are generated when the crossbar override starts.

The crossbar override mode provides several options, defined independently for each output with the COFMA[2:0] and COFMB[2:0] bitfields in the HRTIM_COFExR registers.

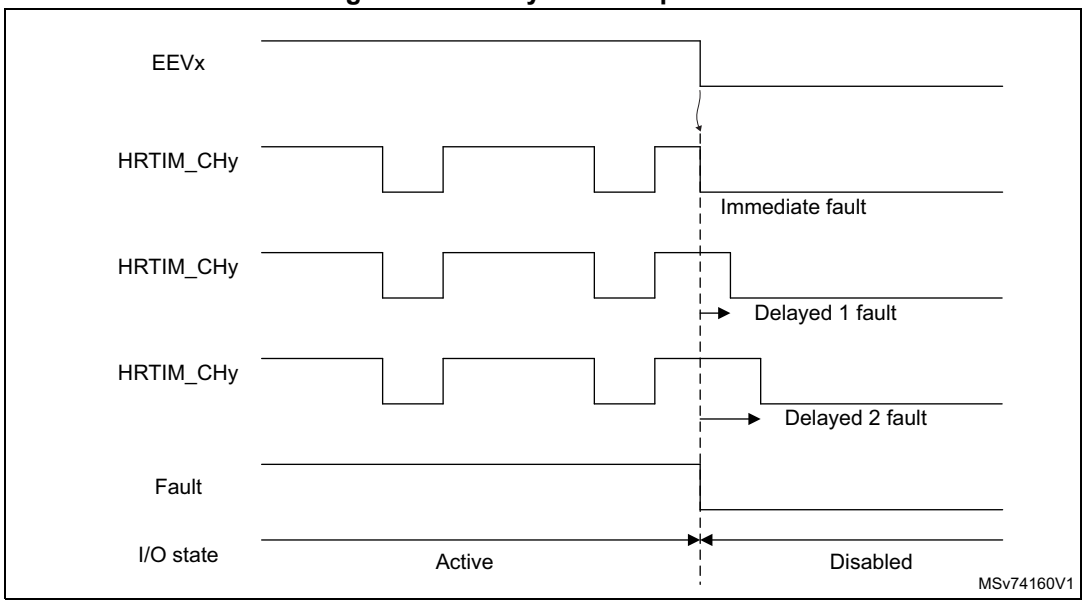
Delayed fault operation

This mode is enabled with COFMx[2:0] = 000 and 001. When the crossbar override mode is triggered, the output level is maintained as it is and forced to the inactive state after a programmable duration. The purpose of this mode is have a delayed shutdown sequence for multi-level inverters. 2 delays can be selected for a given output: Delay 1 with COFMx[2:0] = 000 or Delay 2 with COFMx[2:0] = 001, as shown on *Figure 446*.The delay 1/2 durations are defined with the DLYxFLTD[8:0] bitfields in the HRTIM_CODExR register.

This mode can also be used with deadtime disabled (DTEN=0).

Warning: It is mandatory to program a Delay 2 duration above the Delay 1 duration: DLY2FLTD[8:0] > DLY1FLTD[8:0].

Figure 446. Delayed fault operation

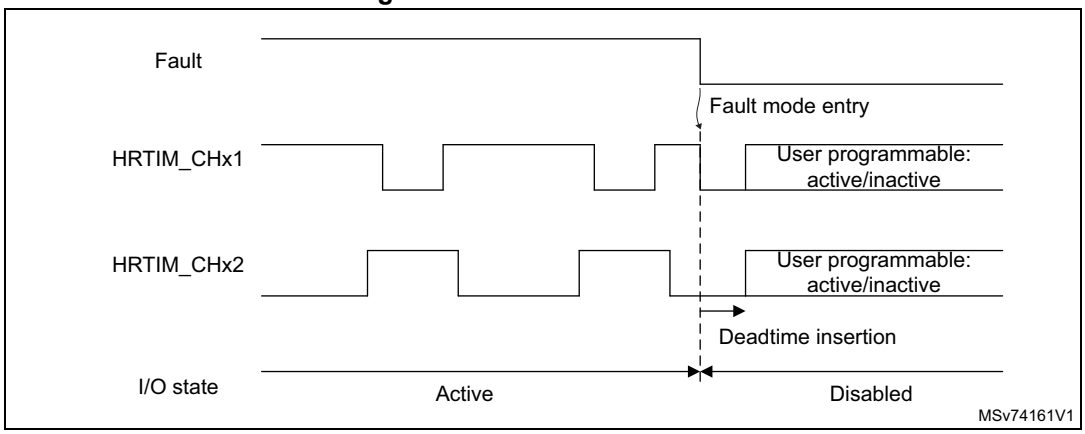


Deadtime on Fault

This mode is enabled with COFMx[2:0] = 010 and 011. When the crossbar override mode is triggered, the output level is forced to an active or inactive state, as shown on *Figure 447* below. As the crossbar override logic is placed prior to the deadtime insertion logic, this mechanism guarantees that an output can be force active without any risk of deadtime violation. If both outputs are forced to the active state, both output 1 and output 2 are forced inactive.

This mode cannot be used with deadtime disabled (DTEN = 0).

Figure 447. Deadtime on Fault

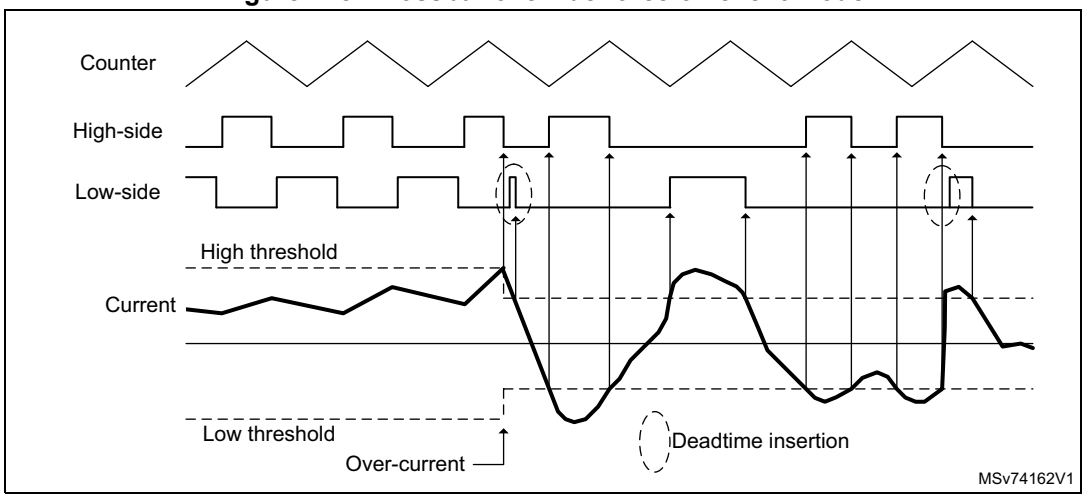


Force on event

This mode is enabled with COFMx[2:0] = 100. When the crossbar override mode is triggered, the output level is forced active on the event selected in the EEVxFRCA and EEVxFRCB bits, in the HRTIM_COFExR register, as shown on *Figure 448* below.

This mode cannot be used with deadtime disabled (DTEN = 0).

Figure 448. Crossbar override force on event mode



Status bits and interrupts

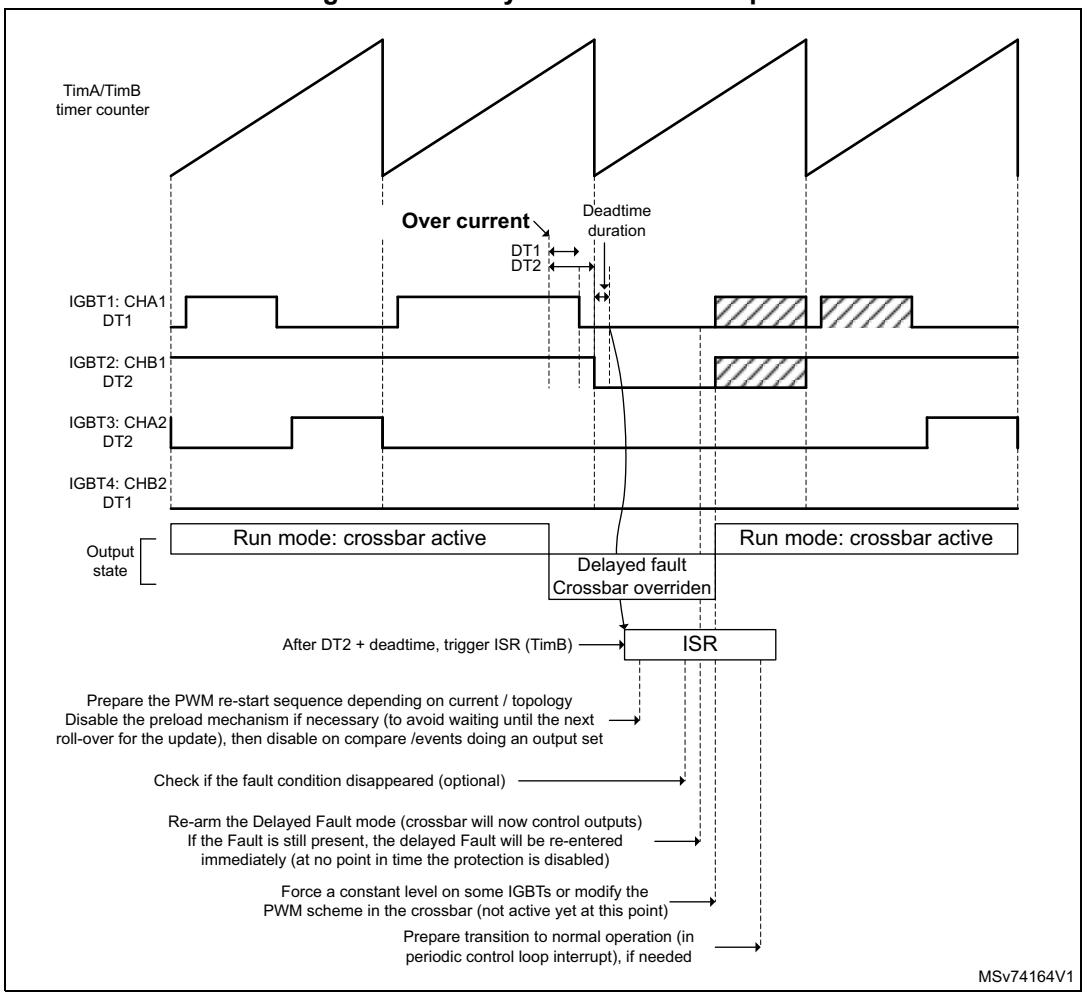
Two status bits are available for software polling, in the HRTIM_TIMxISR register. The COS bit indicates if the crossbar override mode is active, or if the regular crossbar operation is active or was resumed.The COTS bit indicates if the crossbar override trigger is active or not, and can be polled before rearming the crossbar override mode. Both bits are only status bits.

Two interrupts are available. The crossbar override trigger interrupt indicates that regular crossbar operation was interrupted and crossbar override mode was started (COT interrupt flag in HRTIM_TIMxISR). The crossbar override ream delay elapsed interrupt (CORDE interrupt flag in HRTIM_TIMxISR) indicates that the regular crossbar mode can be rearmed. This flag is raised once the Delay 2 period plus the deadtime duration is elapsed, to indicate that the crossbar override delayed fault entry sequence is completed before rearming.

Any rearming attempt before CORDE interrupt is ignored. The *Figure 449* below gives an example of a delayed fault exit sequence [To be completed and reworked].

This CORDE interrupt is generated only when at least on of the one of the 2 channels is configured to Delayed fault force modes (COFMx[2:0] = 000 and 001).

Figure 449. Delayed Fault exit example

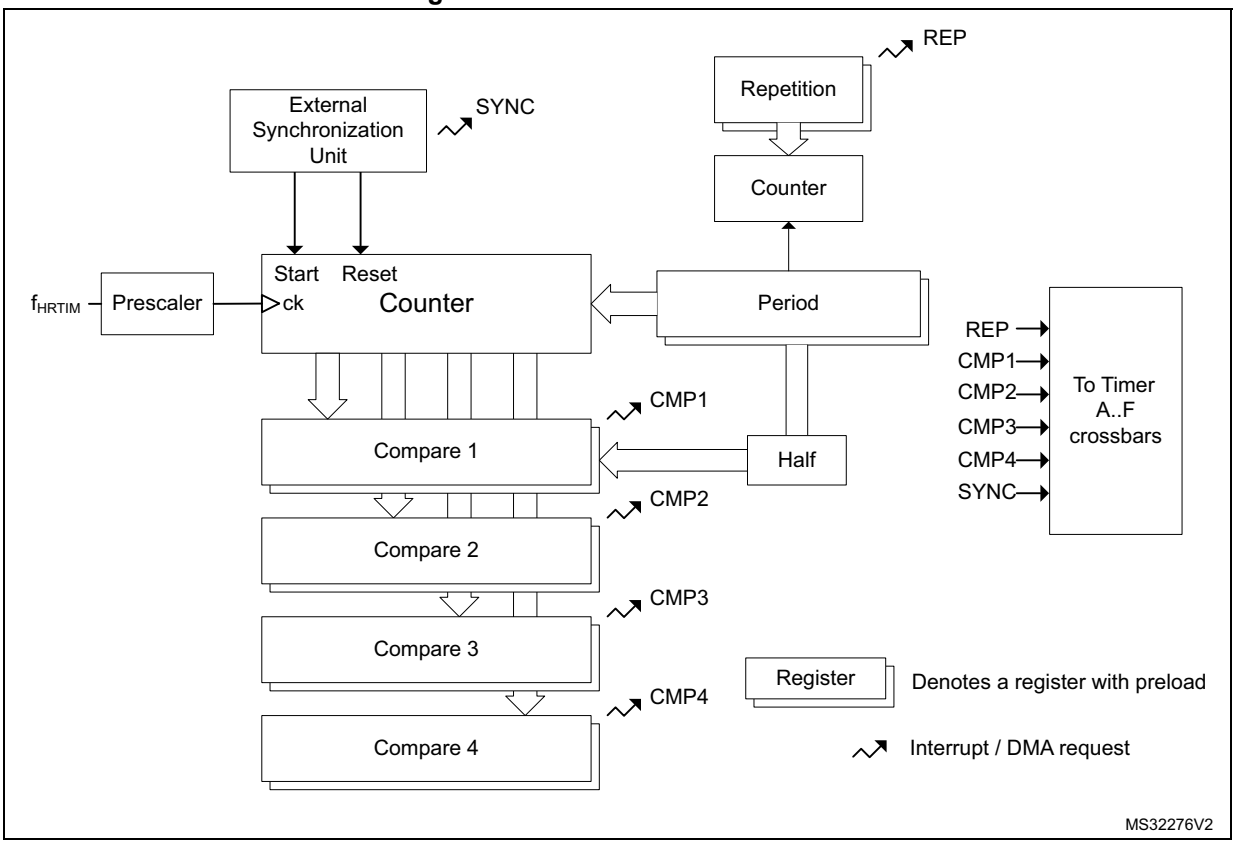


33.3.5 Master timer

The main purpose of the master timer is to provide common signals to the 6 timing units, either for synchronization purpose or to set/reset outputs. It does not have direct control over any outputs, but still can be used indirectly by the set/reset crossbars.

Figure 450 provides an overview of the master timer.

Figure 450. Master timer overview



The master timer is based on the very same architecture as the timing units, with the following differences:

- It does not have outputs associated with, nor output related control
- It does not have its own crossbar unit, nor push-pull or deadtime mode
- It can only be reset by the external synchronization circuitry
- It does not have a capture unit, nor the autodelayed mode
- It does not include external event blanking and windowing circuitry
- It has a limited set of interrupt/DMA requests: compare 1..4, repetition, register update and external synchronization event.

The master timer control register includes all the timer enable bits, for the master and timer A..F timing units. This allows to have all timer synchronously started with a single write access.

It also handles the external synchronization for the whole HRTIM timer (see *Section 33.3.19: Synchronizing the HRTIM with other timers or HRTIM instances*), with both MCU internal and external (inputs/outputs) resources.

Master timer control registers are mapped with the same offset as the timing units' registers.

33.3.6 Up-down counting mode

The HRTIM is natively designed with up-counters. It offers however an operating mode with up-down counters, also called center-aligned mode.

This mode is enabled using the UDM bit in the HRTIM_TIMxCR2 register. This bit must not be changed once the timer is operating (TxEN bit set). It is only available for the TIMA..F. The master timer only works in up-counting mode.

Not all HRTIM features are supported in up-down counting. This section details the functional differences versus up-counting mode.

The period in HRTIM_PERxR must be preloaded (or static) in up-down mode. It can be updated only on period event or on counter reset.

The set/reset crossbar programming differs as follows:

The events coming from the timing units are setting/resetting the outputs depending on the counter up/down direction:

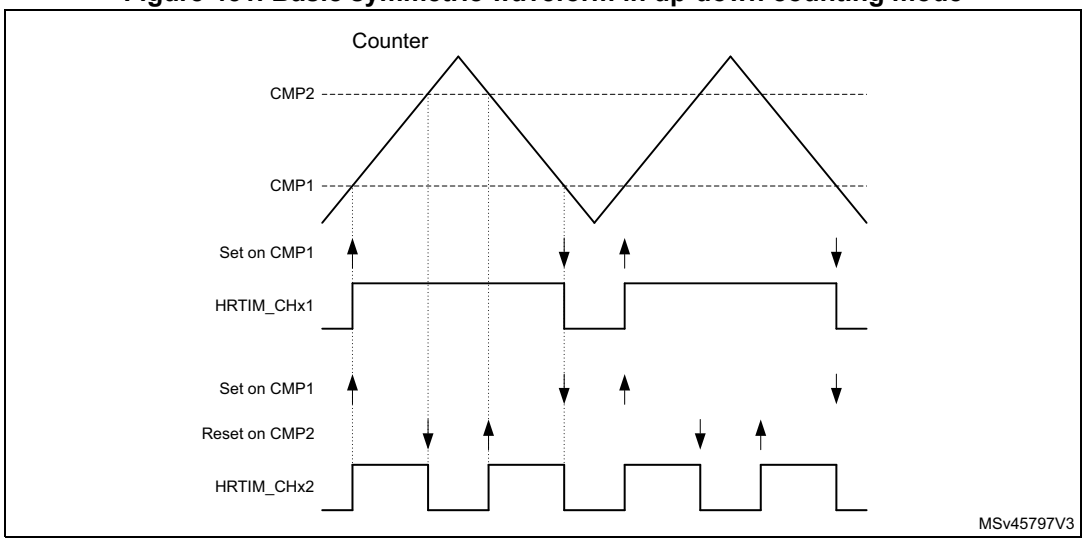
- If the event is enabled in the HRTIM_SETxyR register, it sets the output during up-counting and reset it during down-counting.
- If the event is enabled in the HRTIM_RSTxyR register, it resets the output during up-counting and set it during down-counting.
- If the events are enabled both in HRTIM_SETxyR and HRTIM_RSTxyR registers, the output toggles.

This applies to:

- The timing unit: period, compare 1..4, register update (6 events)
- The master timer: period, compare 1..4, HRTIM synchronization (6 events)
- All other timing units (e.g. timer B..F for timer A): TIMEVNT1..9 (9 events described in *Table 339*).

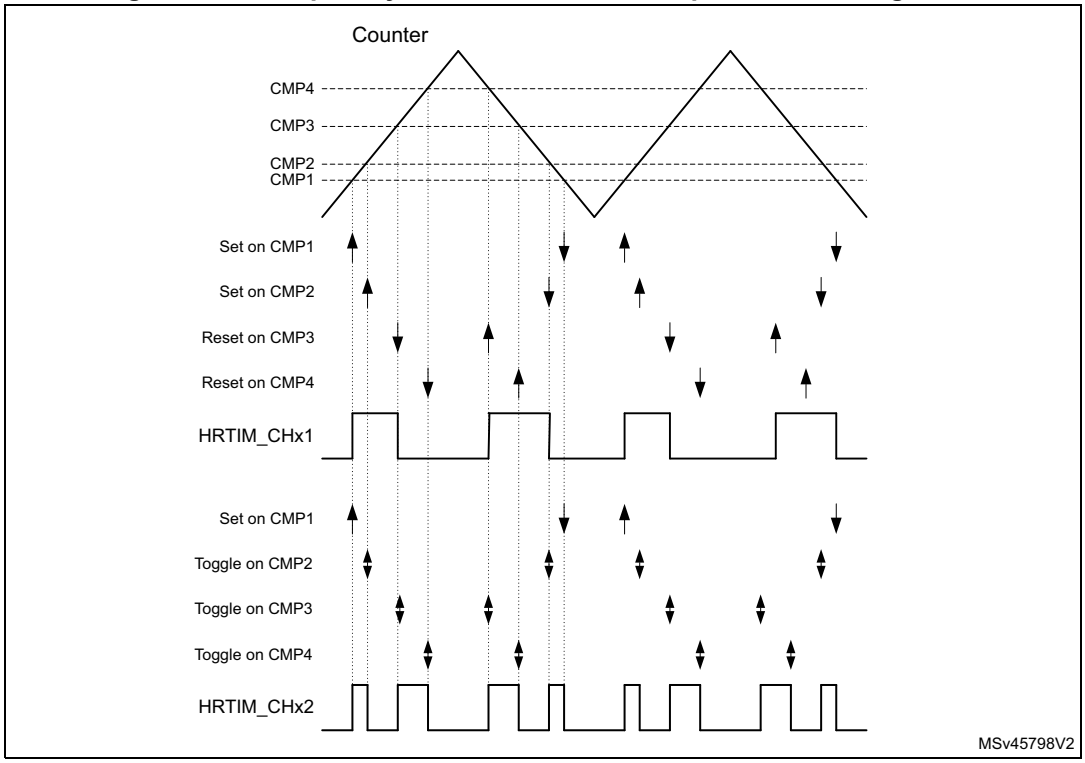
The *Figure 451* below shows how to generate basic waveforms.

Figure 451. Basic symmetric waveform in up-down counting mode



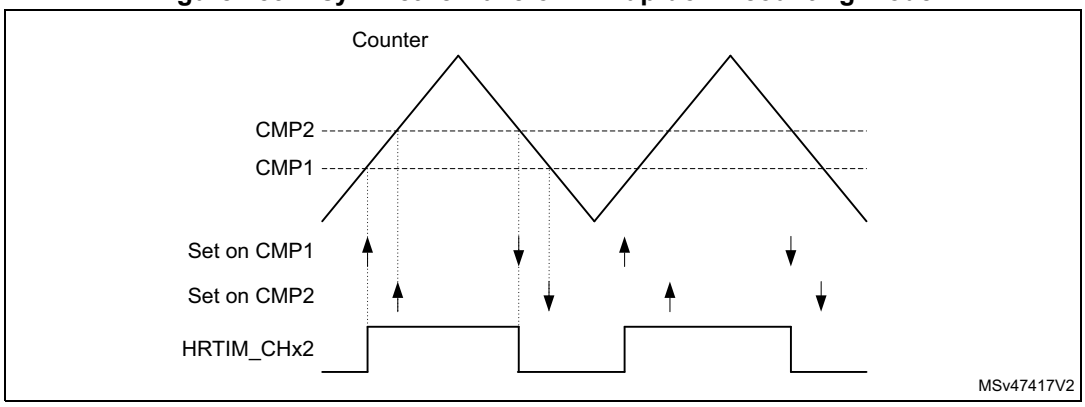
The *Figure 452* below shows how to generate some more complex waveforms, using the 4 available compare units and the toggle mode.

Figure 452. Complex symmetric waveform in up-down counting mode



The *Figure 453* shows how to generate an asymmetric waveform. In this case, it must be noticed that it is necessary to have compare 2 value greater than compare 1 value for the waveform to be asymmetric.

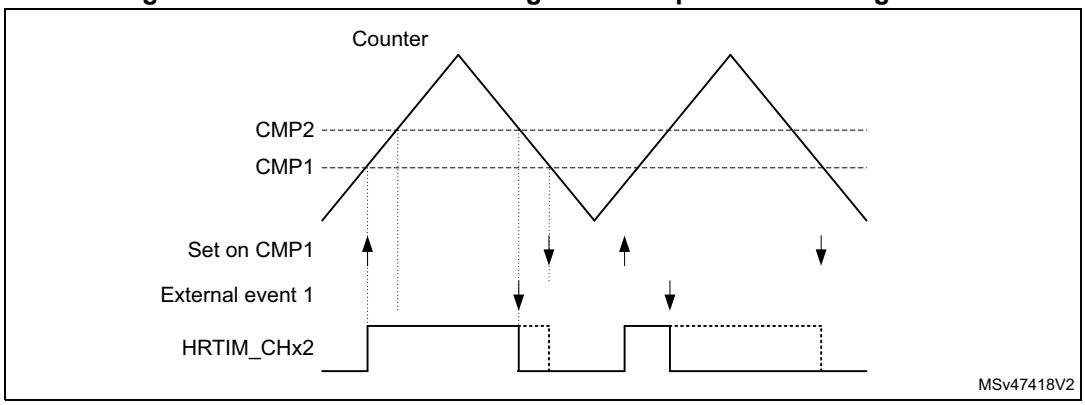
Figure 453. Asymmetric waveform in up-down counting mode



Note: For asymmetric operation, it is required that CMP2 > CMP1.

The behavior of the software forcing bits and external events EXTEVNTA..J is identical for up-only and up-down counting mode. The *Figure 454* below shows how a pulse can be shorten in response to external events.

Figure 454. External event management in up-down counting mode



The up-down counting mode is available for both continuous and single-shot (retriggerable and non-retriggerable) operating modes. A reset causes the counter to re-start from 0. The *Figure 455* below shows the counter behavior in single-shot retriggerable mode, for TIMB.

Figure 455. Interleaved up-down counter operation

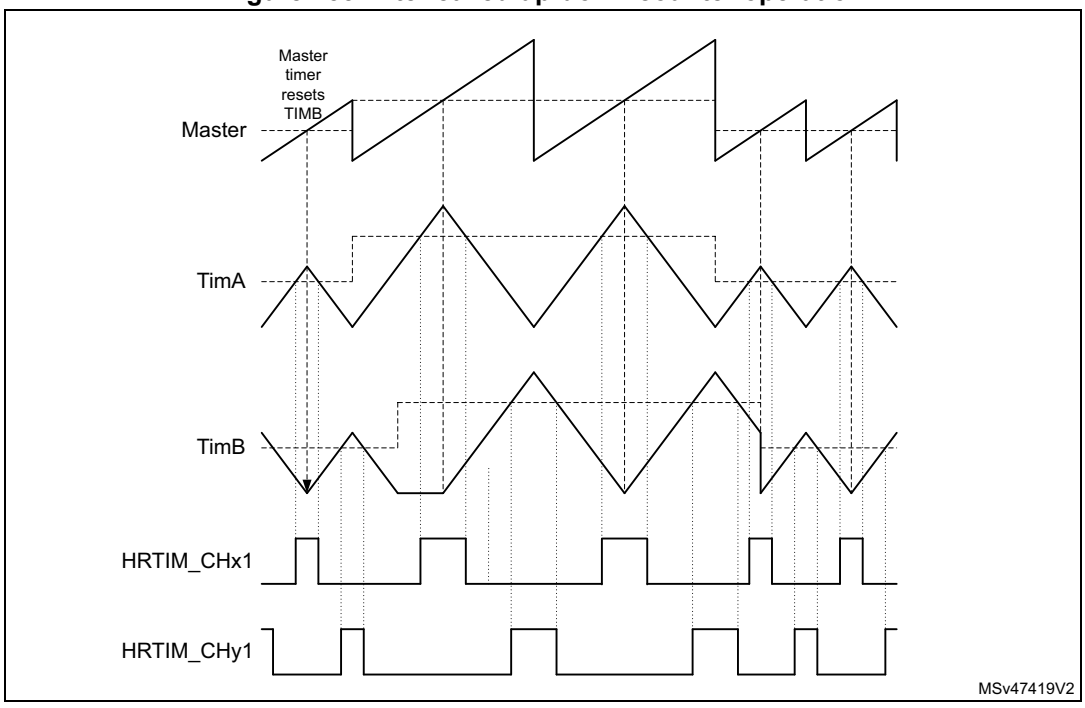
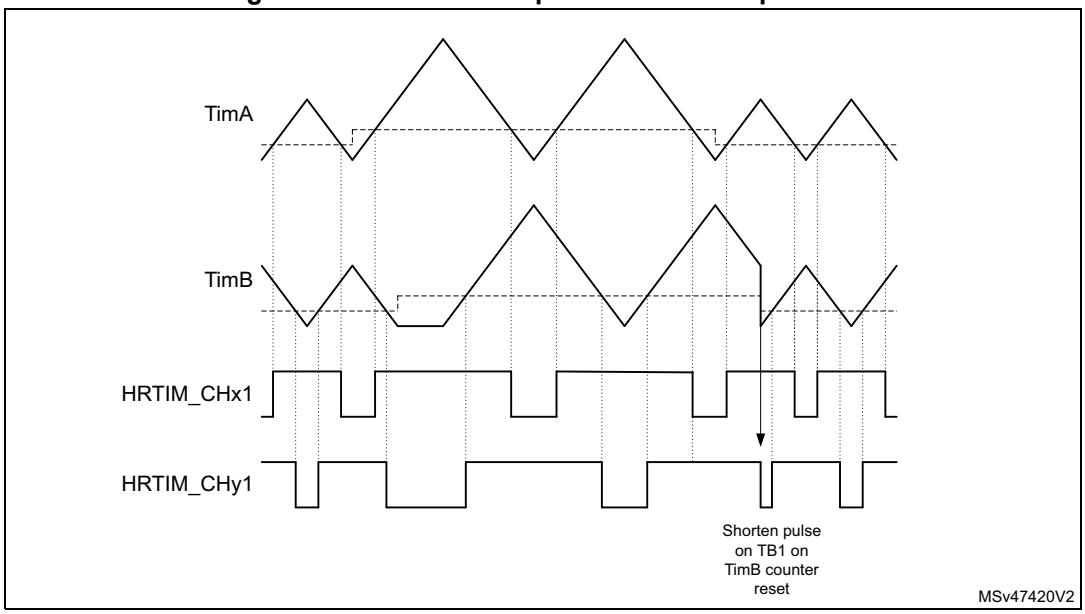


Figure 456. Interleaved up-down counter operation



Note: In up-down counting mode, the compare values must be 3 periods of the f_{HRTIM} clock below the period value (TIMx_PER - 0x60 if CKPSC[2:0] = 0, TIMx_PER - 0x30 if CKPSC[2:0] = 1, TIMx_PER - 0x18 if CKPSC[2:0] = 2,...). This applies for the compare events generated inside the timing unit. For compare events generated in other timing units, it is mandatory to avoid any event occurring within less than 1 period of the f_{HRTIM} clock of a counter direction change (counter at 0, period event or counter reset).

The following features are supported in up-down counting mode:

- Half mode
- Deadtime insertion
- Push-pull mode, alternance push-pull done on when counter = 0 (see *Figure 457*).
- Delayed Idle
- Burst mode
- PWM mode with “greater than” comparison (see *Figure 458*).

Figure 457. Push-pull up-down mode example

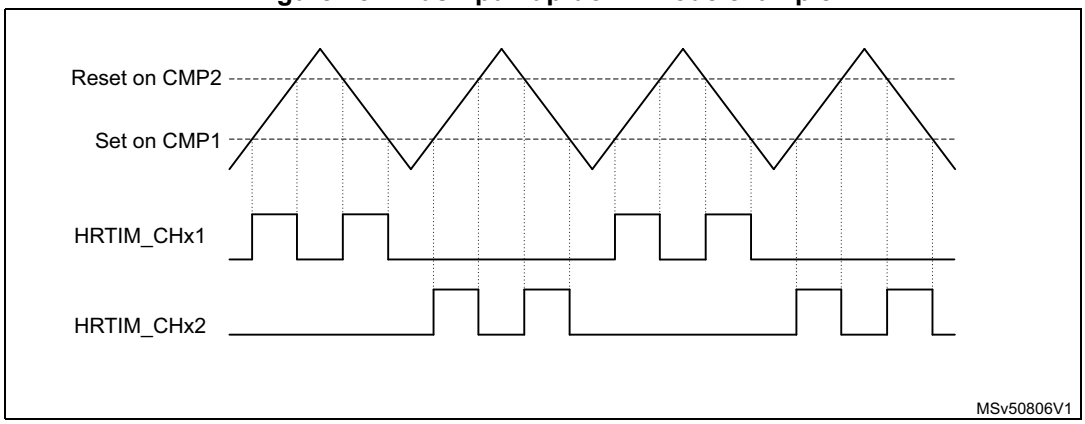
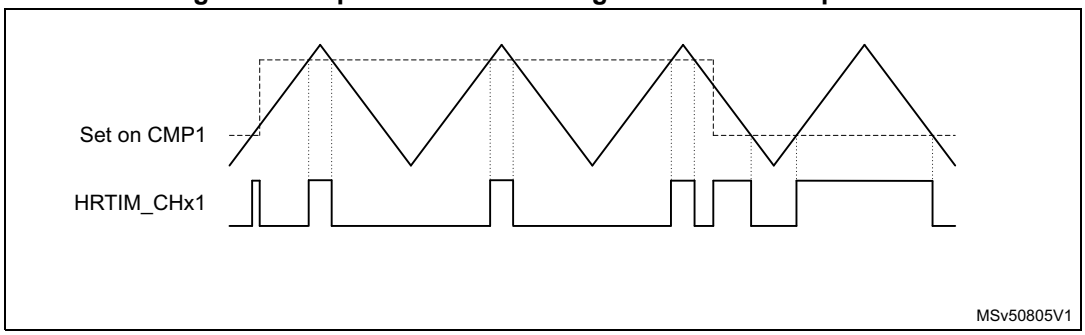


Figure 458. Up-down mode with “greater than” comparison



Caution: The following features are not supported in up-down counting mode:

- Auto-delayed mode
- Balanced Idle
- Triggered-half mode

The capture function is supported with the following differences:

- The capture value is referred to counting start, when up-counting
- The capture value is referred to the PER event when down counting
- The bit 16 of the capture register holds the counting direction status.

The counter roll-over event is defined differently in-up-down counting mode to support various operating condition. It can be either generated:

- When the counter is equal to 0 (“valley” mode)
- When the counter reaches the period value set in the HRTIM_PERxR (“crest” mode)
- When both conditions are met (either 0 or HRTIM_PERxR value).

This event is used for multiple purposes in the HRTIM. The generation mode (valley, crest or both) can be programmed individually depending on the destination. *Table 343* summarizes the use cases and associated roll-over mode (xxROM[1:0]) programming bits in the HRTIM_TIMxCR2 register.

Table 343. Roll-over event destination and mode programming

Roll-over event use	Programming bits
Output set/reset	OUTROM[1:0]
Register content update trigger (transfer from preload to active)	ROM[1:0]
IRQ and/or DMA request trigger	ROM[1:0]
Burst mode clock source and /or burst start trigger	BMROM[1:0]
ADC trigger (see *Section : ADC postscaler* for details)	ADROM[1:0]
External event filtering	ROM[1:0]
Repetition counter decrement	ROM[1:0]
Fault and event counter	FEROM[1:0]

Note: For events where both reset and roll-over are considered (IRQ/DMA and TxRSTU), the xxROM[1:0] influences both the reset and roll-over generation. The reset event is propagated only when xxROM[1:0] = 00 or 01, regardless of the counting direction.

The roll-over event generation is defined as per following xxROM[1:0] bitfield setting:

- xxROM[1:0] = 00: event generated when any of these three conditions are met: counter = 0, counter = HRTIM_PERxR or counter reset.
- xxROM[1:0] = 01: event generated when the counter is equal to 0 (“valley” mode) or when the counter is reset
- xxROM[1:0] = 10: event generated when the counter reaches the period value set in the HRTIM_PERxR (“crest” mode)

The following events are conditioned by the xxROM[1:0] bit fields settings:

- Period event

  It can trigger output set/reset (as per OUTROM[1:0] setting), ADC conversions (as per ADROM[1:0] settings) and burst mode (as per BMROM[1:0] settings), both in continuous and single mode, when the period is explicitly listed as trigger source (for instance ADC1TAPER bit set in HRTIM_ADC1R). When both reset and roll-over events are enabled (for instance ADC1TARST bit set in HRTIM_ADC1R), the period event is considered only in continuous mode (roll-over event). In single-shot mode, no event is generated.
- Roll-over event

  This is similar to period event, for what about xxxROM[1:0] programming, and only valid in continuous mode. In single-shot mode, no event is generated.
- Compare event

  It triggers ADC conversion (as per ADROM[1:0] setting):
  - During up-counting if ADROM = 10
  - During down-counting if ADROM = 01
  - During both up and down-counting if ADROM = 00

Figure 459 here-below shows a push-pull up-down mode with set on period event and OUTROM[1:0] =10.

Figure 459. Up-down mode with output set on period event, for OUTROM[1:0]=10

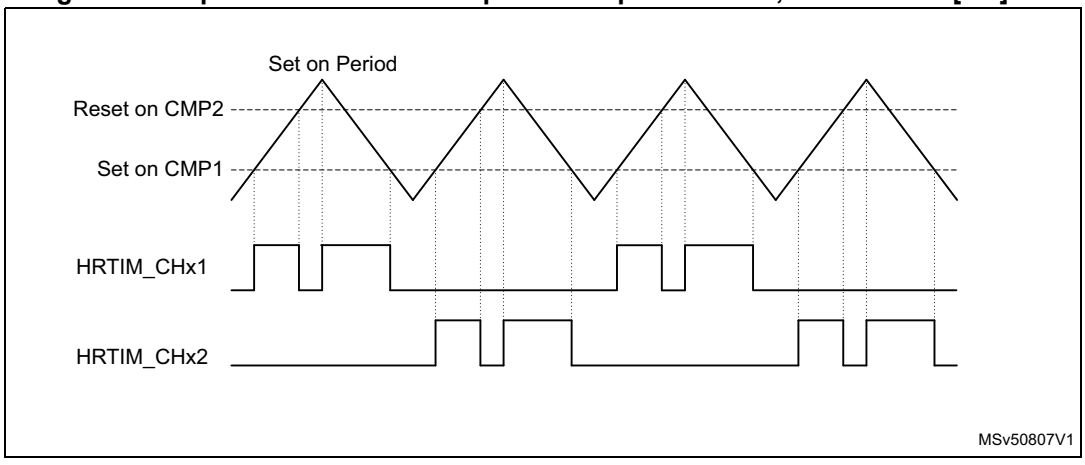
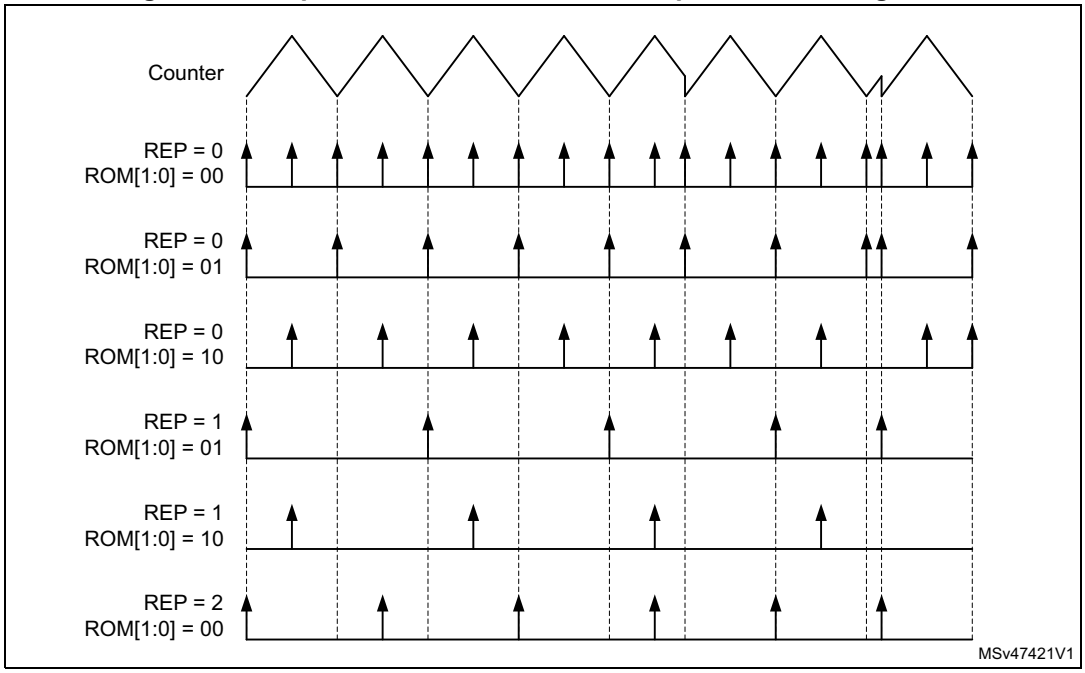


Figure 460 below shows how the repetition counter is decremented in up-down counting mode.

Figure 460. Repetition counter behavior in up-down counting mode



The dual channel DAC triggers are working as for the up-counting mode.

The event blanking and windowing differs so as to have the blanking or windowing done within the output pulse, at a programmable time. The EExFLTR[3:0] codes are depending on the UDM bit setting, as per the *Table 344* below. Whenever the roll-over event is used for blanking or windowing, the ROM[1:0] programming applies for defining when it is generated.

Table 344. EExFLTR[4:0] codes depending on UDM bit setting

EExFLTR[4:0]	Up-counting mode (UDM = 0)	Up/down-counting mode (UDM = 1)
00010	Blanking from counter reset/roll-over to compare 2	Blanking from compare 1 to compare 2, only during the up-counting phase
00100	Blanking from counter reset/roll-over to compare 4	Blanking from compare 3 to compare 4, only during the up-counting phase
01101	Windowing from counter reset/roll-over to compare 2	Windowing from compare 2 to compare 3, only during the up-counting phase
01110	Windowing from counter reset/roll-over to compare 3	Windowing from compare 2 to compare 3, only during the down-counting phase
01111	Windowing from another timing unit: TIMWIN source (see *Table 349* for details)	Windowing from compare 2 during the up-counting phase to compare 3 during the down-counting phase

33.3.7 Set/reset events priorities and narrow pulses management

This section describes how the output waveform is generated when several set and/or reset requests are occurring within 3 consecutive t_{HRTIM} periods.

An arbitration is performed during each t_{HRTIM} period, in 2 steps:

1. For each active event, the desired output transition is determined (set, reset or toggle).
2. A predefined arbitration is performed among the active events, see *Section : Concurrent set requests/Concurrent reset requests*).

When set and reset requests from two different sources occur simultaneously or within the same t_{HRTIM} interval, the reset request has the highest priority, unless the set and reset requests originate from the same timing unit. In this case, the CMPx priority scheme applies. For example, consider CMP2 and CMP4:

- If CMP2 performs a set operation and CMP4 performs a reset operation, the output is reset.
- If CMP2 performs a reset operation and CMP4 performs a set operation, the output is set.

Concurrent set requests/Concurrent reset requests

When multiple sources are selected for a set event, an arbitration is performed when the set requests occur within the same f_{HRTIM} clock period.

In case of multiple requests from adjacent timers (TIMEVNT1..9), the request which occurs first is taken into account. The arbitration is done in 2 steps, depending onthe source (from the highest to the lowest priority): CMP4 ⟶ CMP3 ⟶ CMP2 ⟶ CMP1.

If multiple requests from the master timer occur within the same f_{HRTIM} clock period, a predefined arbitration is applied and a single request is taken into account (from the highest to the lowest priority):

MSTCMP4 → MSTCMP3 → MSTCMP2 → MSTCMP1 → MSTCMPER

When multiple requests internal to the timer occur within the same f_{HRTIM} clock period, a predefined arbitration is applied and the requests are taken with the following priority, whatever the effective timing (from highest to lowest):

CMP4 → CMP3 → CMP2 → CMP1 → PER

Note: Practically, this is of a primary importance when multiple compare events can be simultaneously generated or when using autodelayed compare 2 and compare 4 simultaneously (i.e. when the effective set/reset cannot be determined a priori because it is related to an external event). In this case, the highest priority signal must be affected to the CMP4 event.

Last, the highest priority is given to non timing-related: EXTEVNT1..10, RESYNC (coming from SYNC event if SYNCRSTx or SYNCSTRTx is set or from a software reset), update and software set (SST).

As a summary, in case of simultaneous events, the effective set (reset) event is arbitrated between:

- Any TIMEVNT1..9 event
- A single source from the master (as per the fixed arbitration given above)
- A single source from the timer
- The “non timing-related events”.

The same arbitration principle applies to concurrent reset requests. In this case, the reset request has the highest priority, unless the set and reset requests originate from the same timing unit. In this situation, the CMPx priority scheme applies.

A set or reset event occurring within the prescaler clock cycle is delayed to the next active edge of the prescaled clock (as for a counter reset), even if the arbitration is still performed every t_{HRTIM} cycle.

If a reset event is followed by a set event within the same prescaler clock cycle, the latest event is considered.

33.3.8 External events global conditioning

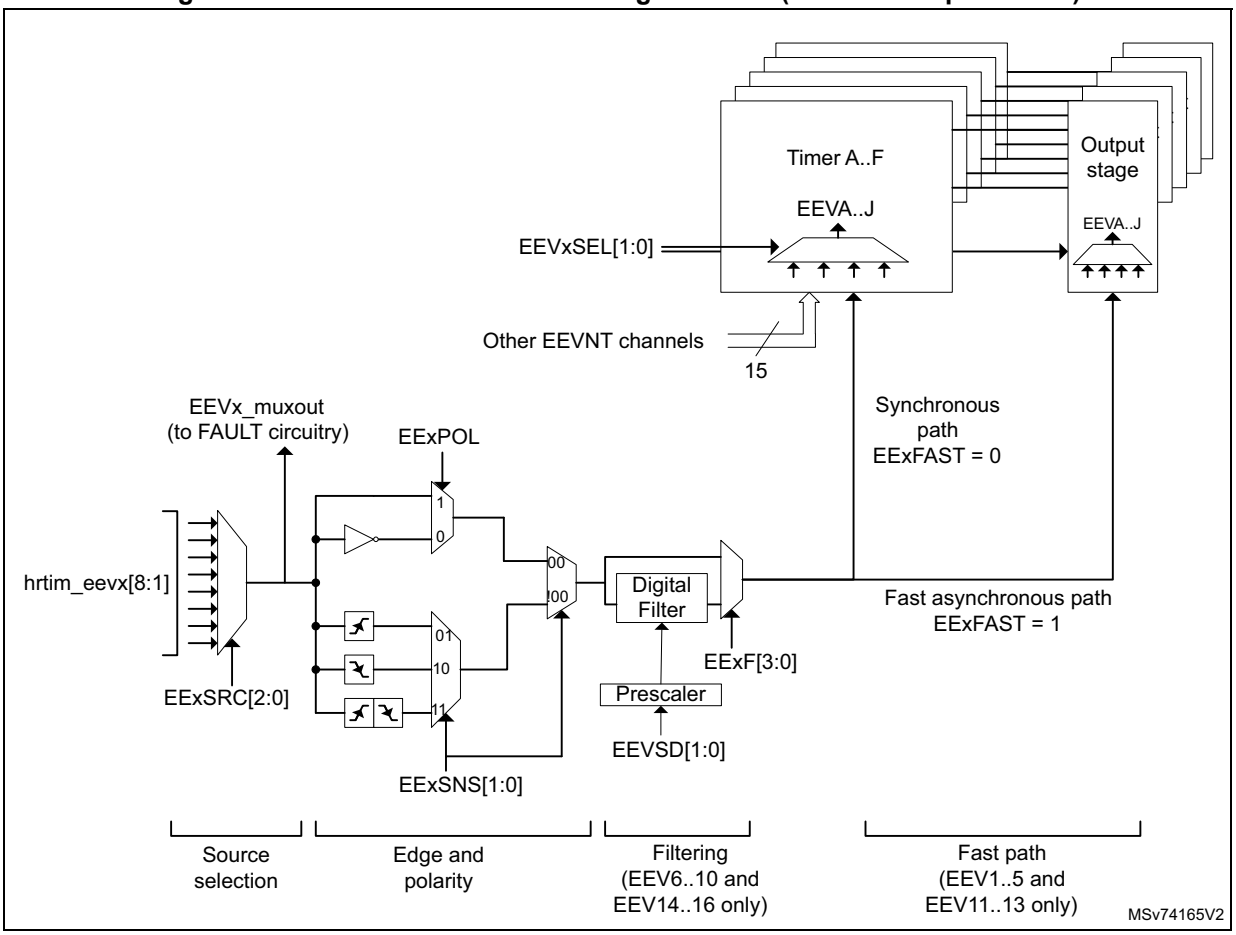
The HRTIM timer can handle events not generated within the timer, referred to as “external event”. These external events come from multiple sources, either on-chip or off-chip:

- built-in comparators,
- digital input pins (typically connected to off-chip comparators and zero-crossing detectors),
- on-chip events for other peripheral (ADC’s analog watchdogs and general purpose timer trigger outputs).

The external events conditioning circuitry allows to select the signal source for a given channel (with a 8:1 multiplexer) and to convert it into an information that can be processed by the crossbar unit (for instance, to have an output reset triggered by a falling edge detection on an external event channel).

Up to 16 external event channels can be conditioned and are available simultaneously for any of the 6 timers. This conditioning is common to all timers, since this is usually dictated by external components (such as a zero-crossing detector) and environmental conditions (typically the filter set-up is related to the applications noise level and signature). *Figure 461* presents an overview of the conditioning logic for a single channel.

Figure 461. External event conditioning overview (1 channel represented)



The 16 external events are initialized using the HRTIM_EECR1, HRTIM EECR2, HRTIM_EECR4, HRTIM_EECR5 and HRTIM EECR7 registers:

- To select up to 8 sources with the EExSRC[2:0] bits,
- To select the sensitivity with EExSNS[1:0] bits, to be either level-sensitive or edge-sensitive (rising, falling or both),
- To select the polarity, in case of a level sensitivity, with EExPOL bit,
- To have a low latency mode, with EExFAST bits (see *Section : Latency to external events*), for external events 1 to 5 and external events 11 to 13.

Note: The external events used as triggers for reset, capture, burst mode, ADC triggers and delayed protection are edge-sensitive even if EESNS bit is reset (level-sensitive selection): if POL = 0 the trigger is active on external event rising edge, while if POL = 1 the trigger is active on external event falling edge.

The external events are discarded as long as the counters are disabled (TxCEN bit reset) to prevent any output state change and counter reset, except if they are used as ADC triggers.

Additionally, it is possible to enable digital noise filters, for external events 6 to 10 using EExF[3:0] bits in the HRTIM_EECR3 register and for external events 14 to 16 using EExF[3:0] bits in the HRTIM_EECR6 register.

A digital filter is made of a counter in which a number N of valid samples is needed to validate a transition on the output. If the input value changes before the counter has

reached the value N, the counter is reset and the transition is discarded (considered as a spurious event). If the counter reaches N, the transition is considered as valid and transmitted as a correct external event. Consequently, the digital filter adds a latency to the external events being filtered, depending on the sampling clock and on the filter length (number of valid samples expected).

The sampling clock is either the f_{HRTIM} clock or a specific prescaled clock f_{EEVS} derived from f_{HRTIM}, defined with EEVSD[1:0] bits in HRTIM_EECR3 register.

Table 345 summarizes the available features associated with each of the 10 external events channels.The features and sources are summarized in *Table 326*.

Table 345. External events features

External event channel	Fast mode	Digital filter	Balanced fault timer A,B,C	Balanced fault timer D,E,F
hrtim_eev1[8:1]	Yes	-	-	-
hrtim_eev2[8:1]	Yes	-	-	-
hrtim_eev3[8:1]	Yes	-	-	-
hrtim_eev4[8:1]	Yes	-	-	-
hrtim_eev5[8:1]	Yes	-	-	-
hrtim_eev6[8:1]	-	Yes	Yes	-
hrtim_eev7[8:1]	-	Yes	Yes	-
hrtim_eev8[8:1]	-	Yes	-	Yes
hrtim_eev9[8:1]	-	Yes	-	Yes
hrtim_eev10[8:1]	-	Yes	-	-
hrtim_eev11[8:1]	Yes	-	-	-
hrtim_eev12[8:1]	Yes	-	-	-
hrtim_eev13[8:1]	Yes	-	-	-
hrtim_eev14[8:1]	-	Yes	-	-
hrtim_eev15[8:1]	-	Yes	-	-
hrtim_eev16[8:1]	-	Yes	-	-

The *Table 346* provides the mapping between EEVA..EEVJ events and external events hrtim_eev1 to hrtim_eev16. This is configured using the EEVxSEL[1:0] bitfield in the HRTIM_EESELxR register.

Table 346. External events selection

External event channel x = A..J	EEVxSEL[1:0]=00	EEVxSEL[1:0]=01	EEVxSEL[1:0]=10	EEVxSEL[1:0]=11
EEVA	hrtim_eev1	hrtim_eev11	hrtim_eev1 AND hrtim_eev2	hrtim_eev1 AND hrtim_eev11
EEVB	hrtim_eev2	hrtim_eev12	hrtim_ee2 AND hrtim_eev3	hrtim_eev2 AND hrtim_eev12

Table 346. External events selection

External event channel x = A..J	EEVxSEL[1:0]=00	EEVxSEL[1:0]=01	EEVxSEL[1:0]=10	EEVxSEL[1:0]=11
EEVC	hrtim_eev3	hrtim_eev13	hrtim_ee3 AND hrtim_eev4	hrtim_eev3 AND hrtim_eev13
EEVD	hrtim_eev4	hrtim_eev14	hrtim_ee4 AND hrtim_eev5	hrtim_eev4 AND hrtim_eev14
EEVE	hrtim_eev5	hrtim_eev15	hrtim_ee5 AND hrtim_eev6	hrtim_eev5 AND hrtim_eev15
EEVF	hrtim_eev6	hrtim_eev16	hrtim_ee6 AND hrtim_eev7	hrtim_eev6 AND hrtim_eev16
EEVG	hrtim_eev7	hrtim_eev11	hrtim_ee7 AND hrtim_eev8	hrtim_eev7 AND hrtim_eev11
EEVH	hrtim_eev8	hrtim_eev12	hrtim_ee8 AND hrtim_eev9	hrtim_eev8 AND hrtim_eev12
EEVI	hrtim_eev9	hrtim_eev13	hrtim_ee9 AND hrtim_eev10	hrtim_eev9 AND hrtim_eev13
EEVJ	hrtim_eev10	hrtim_eev14	hrtim_ee10 AND hrtim_eev11	hrtim_eev10 AND hrtim_eev14

The external events can be combined (hrtim_eevy AND hrtim_eevz) with EEVxSEL[1:0] = 10 or 11.

They can be combined solely if they both are in same mode:

- hrtim_eey (with fast mode option) and hrtim_eevz (with fast mode option), with both EEyFAST = EEzFAST = 0 or EEyFAST = EEzFAST = 1
- hrtim_eey (with digital filter option) and hrtim_eevz (with fast mode option) with EEzFAST = 0

It is not possible to combine hrtim_eey (with digital filter option) and hrtim_eevz (with fast mode option) with EEzFAST = 1.

When EEVA..EEVJ is resulting from a combination of 2 external events (EEVxSEL[1:0] = 10 or 11), at least one of the 2 inputs must be level sensitive. Having a level sensitive input combined with an edge sensitive input results in an edge sensitive combined signal. This gives the possibility to let an external event be generated depending on the level of an other external event (AND function).

Latency to external events

The external event conditioning gives the possibility to adjust the external event processing time (and associated latency) depending on performance expectations:

- A regular operating mode, in which the external event is resampled with the clock before acting on the output crossbar. This adds some latency but gives access to all

crossbar functionalities. It enables the generation of an externally triggered high-resolution pulse.

- A fast operating mode, in which the latency between the external event and the action on the output is minimized. This mode is convenient for ultra-fast over-current protections, for instance.

EExFAST bits in the HRTIM_EECR1 and HRTIM_EECR4 registers allow to define the operating for channels 1 to 5 and channels 11 to 13. This influences the latency and the jitter present on the output pulses, as summarized in the table below.

Table 347. Output set/reset latency and jitter versus external event operating mode

EExFAST	Response time latency	Response time jitter	Jitter on output pulse (counter reset by ext. event)
0	5 to 6 cycles of f_{HRTIM} clock	1 cycles of f_{HRTIM} clock	No jitter, pulse width maintained with high-resolution
1	Minimal latency (depends whether the comparator or digital input is used)	Minimal jitter	1 cycle of f_{HRTIM} clock jitter pulse width resolution down to t_{HRTIM}

The EExFAST mode is only available with level-sensitive programming (EExSNS[1:0] = 00); the edge-sensitivity cannot be programmed.

It is possible to apply event filtering to fast external events (both blanking and windowing with EExFLTR[3:0] ≠ 0000, see *Section 33.3.9*). In this case, EExLTCHx bit must be reset: the postponed mode is not supported, neither the windowing timeout feature.

Note: The external event configuration (source and polarity) must not be modified once the related EExFAST bit is set.

A fast external event cannot be used to toggle an output: if must be enabled either in HRTIM_SETxyR or HRTIM_RSTxyR registers, not in both.

When a set and a reset event - from 2 independent fast external events - occur simultaneously, the reset has the highest priority in the crossbar and the output becomes inactive.

When EExFAST bit is set, the output cannot be changed during the 11 f_{HRTIM} clock periods following the external event.

Figure 462 and *Figure 463* give practical examples of the reaction time to external events, for output set/reset and counter reset.

Figure 462. Latency to external events (counter reset and output set)

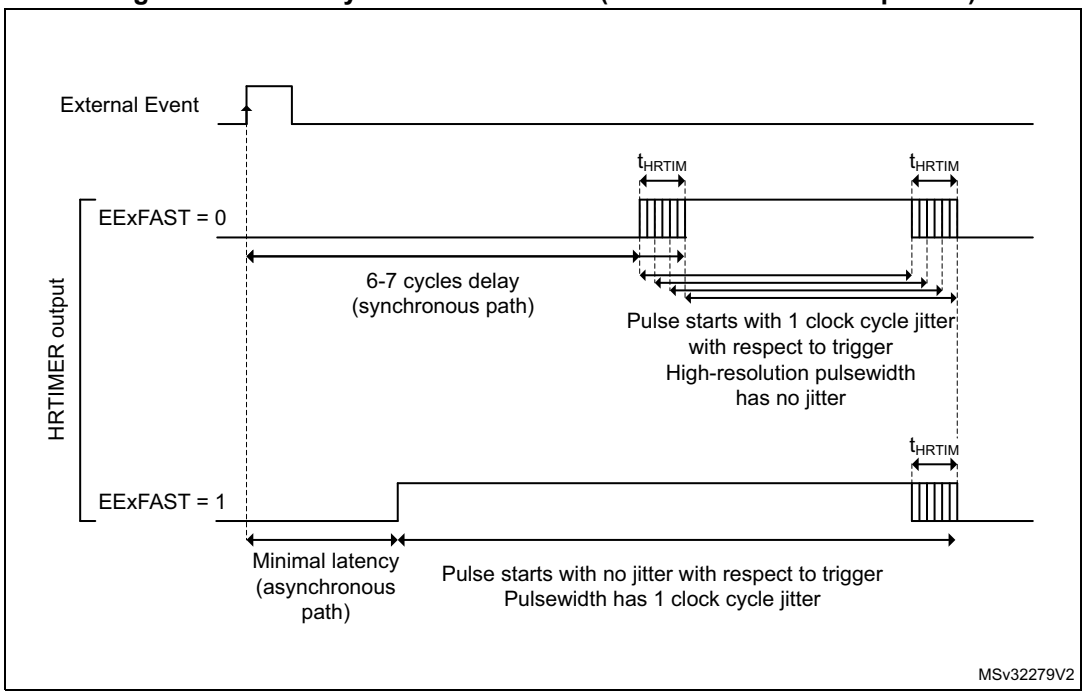
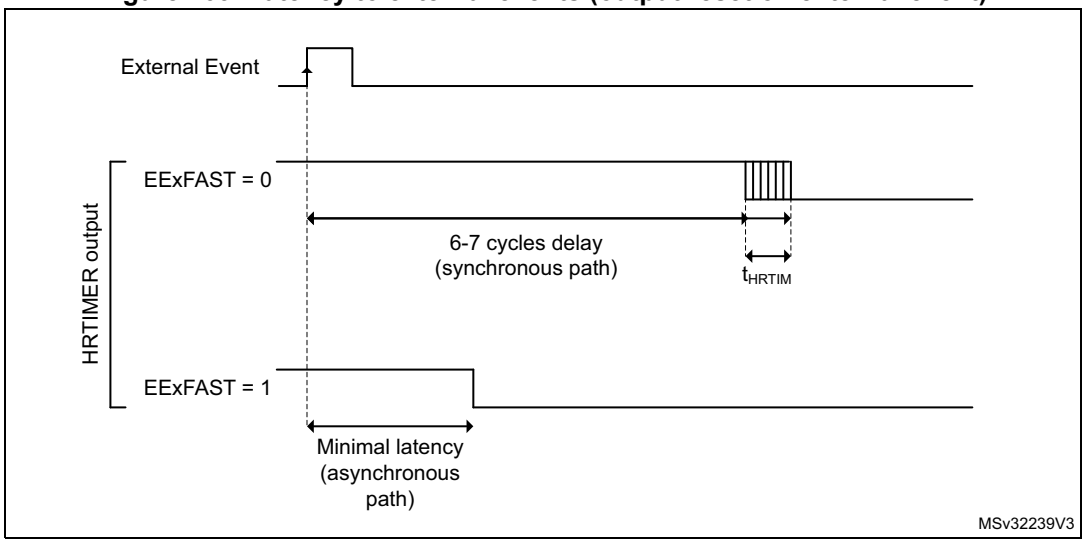


Figure 463. Latency to external events (output reset on external event)



33.3.9 External event filtering in timing units

Once conditioned, the 10 external events are available for all timing units.

They can be used directly and are active as soon as the timing unit counter is enabled (TxCEN bit set).

They can also be filtered to have an action limited in time, usually related to the counting period. Two operations can be performed:

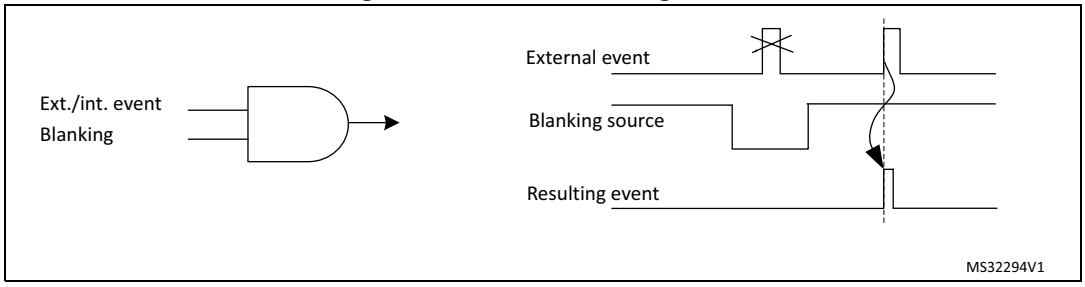
- Blanking, to mask external events during a defined time period,
- Windowing, to enable external events only during a defined time period.

These modes are enabled using HRTIM_EExFLTR[3:0] bits in the HRTIM_EEFxR1 and HRTIM_EEFxR2 registers. Each of the 6 timer A..F timing units has its own programmable filter settings for the 10 external events.

Blanking mode

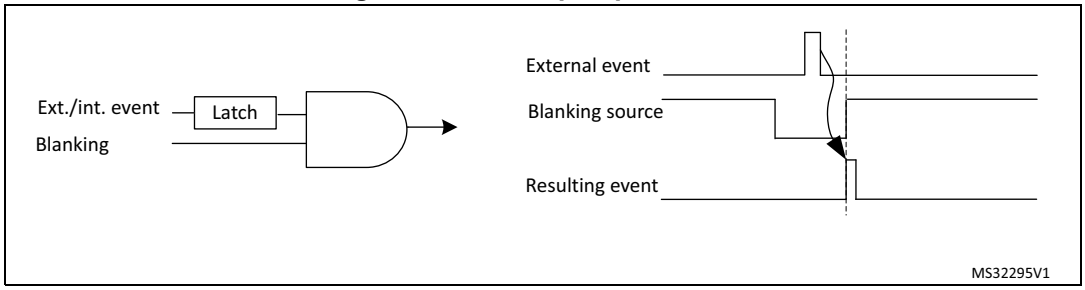
In event blanking mode (see *Figure 464*), the external event is ignored if it happens during a given blanking period. This is convenient, for instance, to avoid a current limit to trip on switching noise at the beginning of a PWM period. This mode is active for EExFLTR[3:0] bitfield values ranging from 0001 to 1100.

Figure 464. Event blanking mode



In event postpone mode, the external event is not taken into account immediately but is memorized (latched) and generated as soon as the blanking period is completed, as shown on *Figure 465*. This mode is enabled by setting EExLTCH bit in HRTIM_EEFxR1 and HRTIM_EEFxR2 registers.

Figure 465. Event postpone mode



The blanking signal comes from several sources:

- the timer itself: the blanking lasts from the counter reset to the compare match (EExFLTR[3:0] = 0001 to 0100 for compare 1 to compare 4). In up/down mode (UDM bit set to 1), the counter reset event is defined as per the ROM[1:0] bit setting.
- from other timing units (EExFLTR[3:0] = 0101 to 1100): the blanking lasts from the selected timing unit counter reset to one of its compare match, or can be fully programmed as a waveform on Tx2 output. In this case, events are masked as long as the Tx2 signal is inactive (it is not necessary to have the output enabled, the signal is taken prior to the output stage).

The EEXFLTR[3:0] configurations from 0101 to 1100 are referred to as TIMFLTR1 to TIMFLTR8 in the bit description, and differ from one timing unit to the other. *Table 348* gives the 8 available options per timer: CMPx refers to blanking from counter reset to compare match, Tx2 refers to the timing unit TIMx output 2 waveform defined with HRTIM_SETx2 and HRTIM_RSTx2 registers. For instance, timer B (TIMFLTR6) is timer C output 2 waveform.

Table 348. Filtering signals mapping per timer

Source		Timer A				Timer B				Timer C				Timer D				Timer E				Timer F			
		CMP1	CMP2	CMP4	TA2	CMP1	CMP2	CMP4	TB2	CMP1	CMP2	CMP4	TC2	CMP1	CMP2	CMP4	TD2	CMP1	CMP2	CMP4	TE2	CMP1	CMP2	CMP4	TF2
Destination	Timer A	-				1	-	2	3	4	-	5	-	7	-	-	-	-	8	-	-	6	-	-	-
	Timer B	1	-	2	3	-				4	5	-	-	-	7	-	-	8	-	-	-	-	6	-	-
	Timer C	-	1	-	-	2	-	3	-	-				5	-	6	7	-	-	8	-	4	-	-	-
	Timer D	1	-	-	-	-	2	-	-	3	4	-	5	-				6	-	7	-	-	-	8	-
	Timer E	-	1	-	-	2	-	-	-	3	-	-	-	6	-	7	8	-				-	-	4	5
	Timer F	-	-	1	-	-	2	-	-	-	-	3	-	-	4	5	-	6	-	7	8	-			

Figure 466 and *Figure 467* give an example of external event blanking for all edge and level sensitivities, in regular and postponed modes.

Figure 466. External event blanking with edge-sensitive trigger

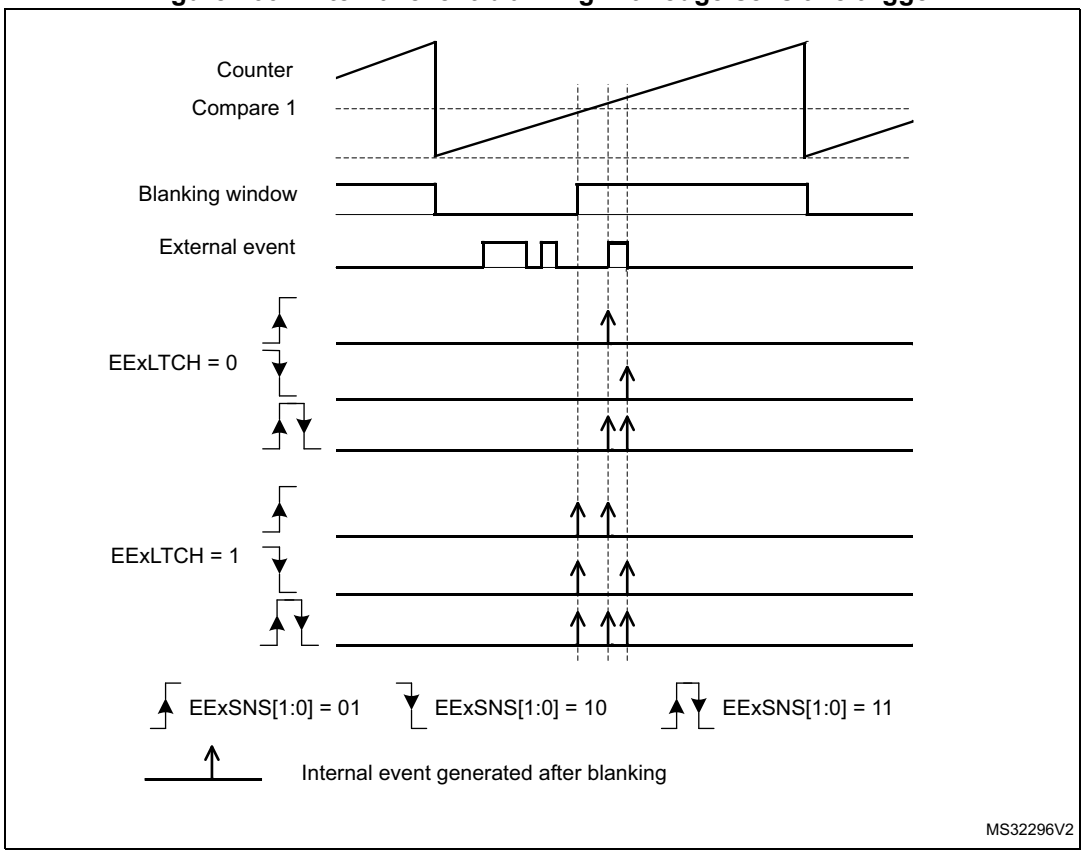
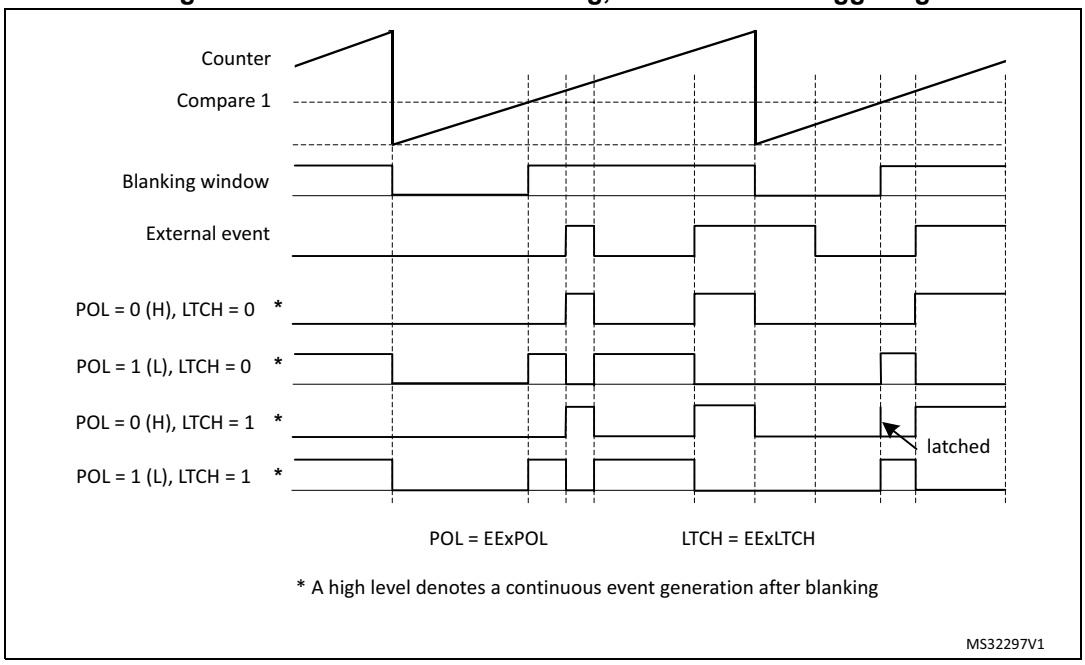


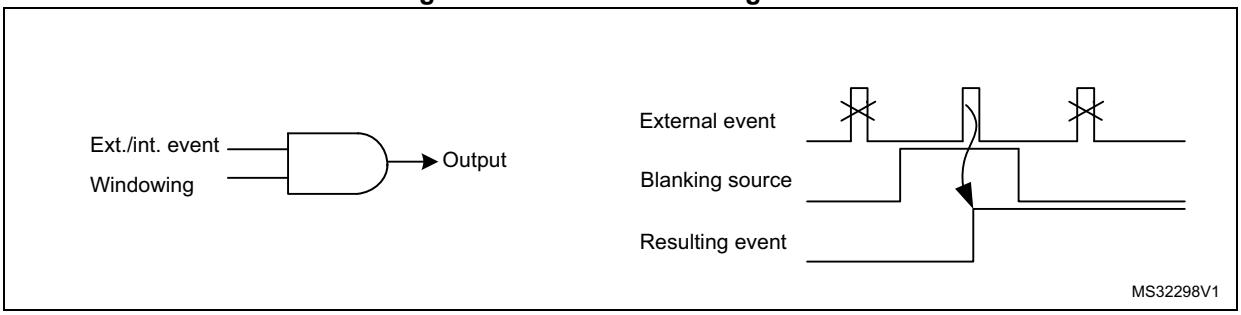
Figure 467. External event blanking, level sensitive triggering



Windowing mode

In event windowing mode, the event is taken into account only if it occurs within a given time window, otherwise it is ignored. This mode is active for EExFLTR[3:0] ranging from 1101 to 1111.

Figure 468. Event windowing mode



EExLTCH bit in EEFxR1 and EEFxR2 registers allows to latch the signal, if set to 1: in this case, an event is accepted if it occurs during the window but is delayed at the end of it.

- If EExLTCH bit is reset and the signal occurs during the window, it is passed through directly.
- If EExLTCH bit is reset and no signal occurs, a timeout event is generated at the end of the window.

A use case of the windowing mode is to filter synchronization signals. The timeout generation allows to force a default synchronization event, when the expected synchronization event is lacking (for instance during a converter start-up).

There are 3 sources for each external event windowing, coded as follows:

- 1101 and 1110: the windowing lasts from the counter reset to the compare match (respectively compare 2 and compare 3). In up/down mode (UDM bit set to 1), the counter reset event is defined as per the ROM[1:0] bit setting.
- 1111: the windowing is related to another timing unit and lasts from its counter reset to its compare 2 match. The source is described as TIMWIN in the bit description and is given in *Table 349*. As an example, the external events in timer B can be filtered by a window starting from timer A counter reset to timer A compare 2.

Table 349. Windowing signals mapping per timer (EEFLTR[3:0] = 1111)

Destination	Timer A	Timer B	Timer C	Timer D	Timer E	Timer F
TIMWIN (source)	Timer B CMP2	Timer A CMP2	Timer D CMP2	Timer C CMP2	Timer F CMP2	Timer E CMP2
TIMWIN2 (source)	Timer B CMP2 to reset/roll-over	Timer A CMP2 to reset/roll-over	Timer D CMP2 to reset/roll-over	Timer C CMP2 to reset/roll-over	Timer F CMP2 to reset/roll-over	Timer E CMP2 to reset/roll-over
TIMWIN3 (source)	Timer B CMP1 to CMP2	Timer A CMP1 to CMP2	Timer D CMP1 to CMP2	Timer C CMP1 to CMP2	Timer F CMP1 to CMP2	Timer E CMP1 to CMP2

Note: The timeout event generation is not supported if the external event is programmed in fast mode.

Figure 469 and *Figure 470* present how the events are generated for the various edge and level sensitivities, as well as depending on EExLTCH bit setting. Timeout events are specifically mentioned for clarity reasons.

Figure 469. External event windowing with edge-sensitive trigger

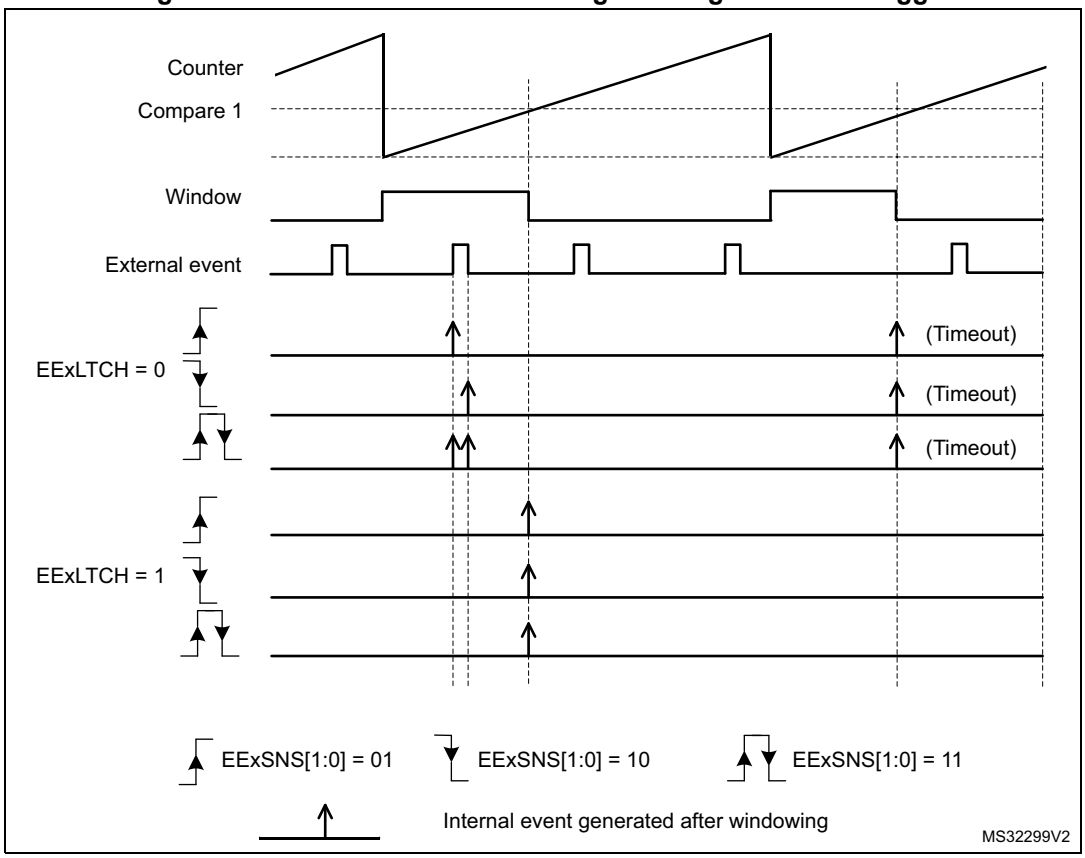
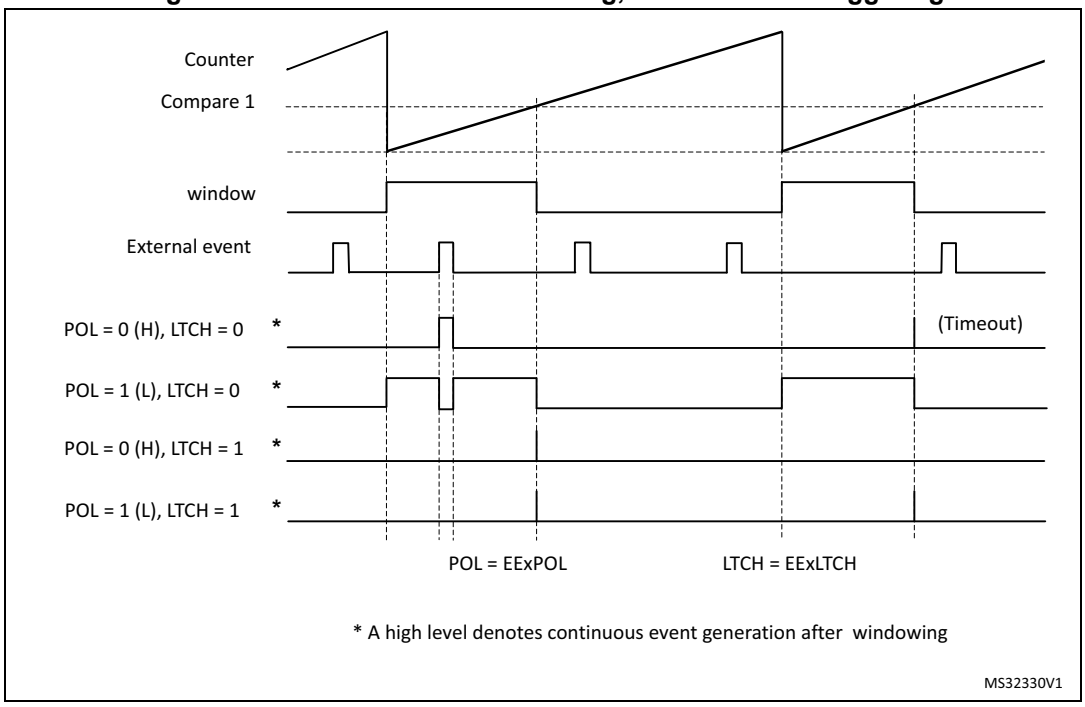


Figure 470. External event windowing, level sensitive triggering

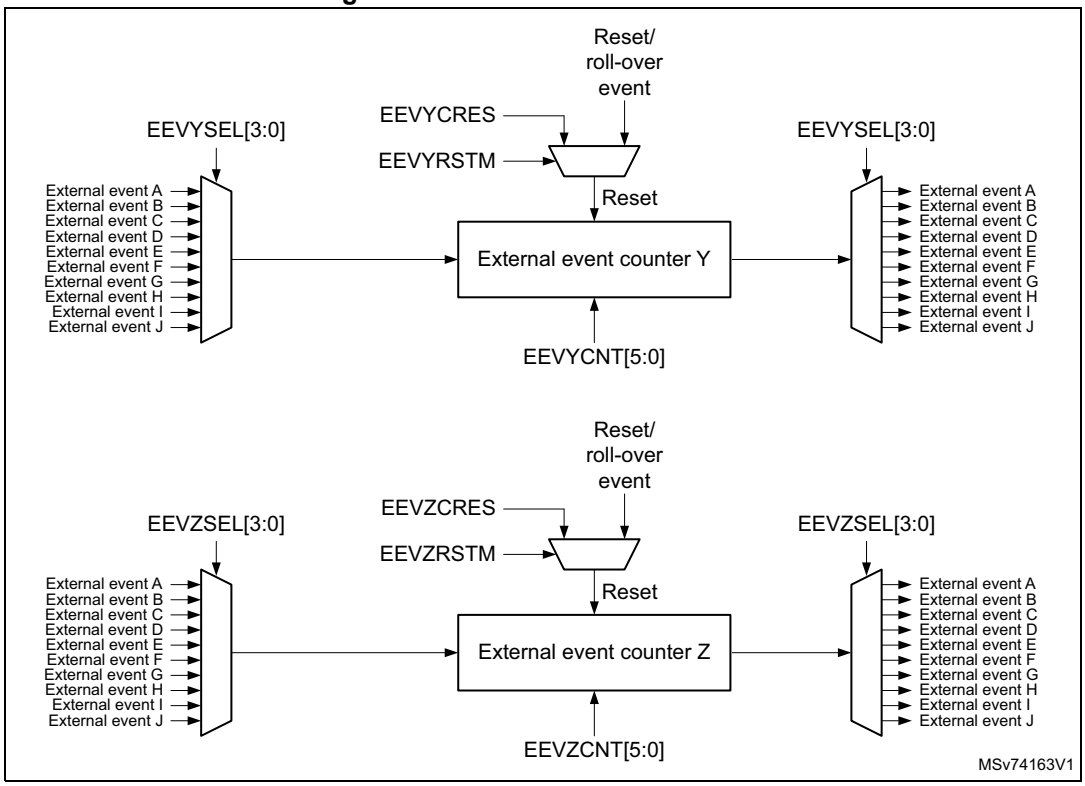


External event counter

Each timing unit also features two external event counters following the filtering unit, typically for valley skipping implementation.

The circuitry allows to filter any of the 10 external events filtered, as shown on *Figure 471*.

Figure 471. External event counters



The counters Y and Z are enabled using the EEVxCE bit in the HRTIM_EEFxR3 register. This mode is only valid for edge-sensitive external events (EExSNS[1:0] bitfield = 01, 10 or 11), when EEVA..EEVJ selection is a single external event EEV1..16 (EEVxSEL[1:0] = 00 or 01).

When EEVA..EEVJ is resulting from a combination of 2 external events (EEVxSEL[1:0] = 10 or 11), at least one of the 2 inputs must be level sensitive. Having a level sensitive input combined with an edge sensitive input results in an edge sensitive combined signal. This gives the possibility to let an external event be generated depending on the level of an other external event (AND function).

The external event is propagated to the timer only if the number of active edges is greater or equal to the value programmed in (EEVxCNT[5:0]+1).

Two operating modes are available:

- When the EEVxRSTM bit is reset, the external event counter is reset on each reset/roll-over event: the external event is active only if it appears several times within a given PWM period
- When the EEVxRSTM bit is set, the external event counter is reset only if the event did not on appear during the last PWM period. This a cumulative mode, where the event must occur at least once during multiple PWM period, as shown on *Figure 472* below.

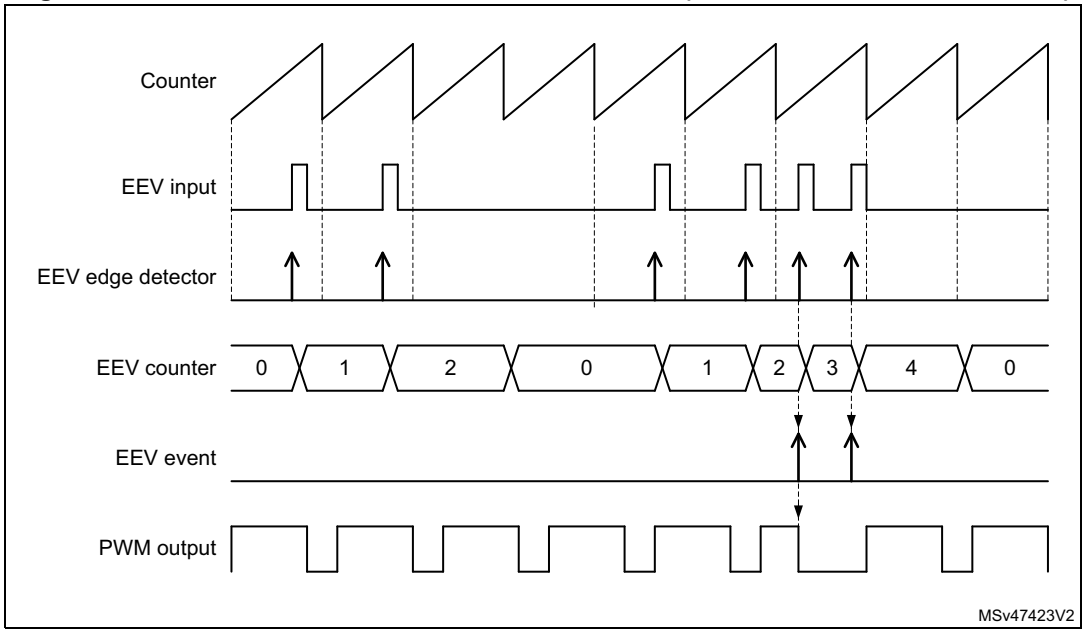
Note: If the 2 external event counters are operating on the same external event (EEVYSEL[3:0] equal to EEVZSEL[3:0]) the external counter Y only is considered.

The external event counter must be enabled after having programmed the counter value (the EEVxCE bit must be set after having written the EEVxCNT[5:0] bits).

Once the counter is enabled, the EEVxCNT[5:0] bits can then be changed on-the-fly at any time. The new value is taken into account on the following reset/rollover event as per the EEVxRSTM bit programming, or after a software reset (EEVxCRES bit set).

The EEVxSEL[3:0] bitfield must not be modified once the EEVxCE bit is set.

Figure 472. External event counter cumulative mode (EEVxRSTM = 1, EEVxCNT = 2)



33.3.10 Delayed protection

The HRTIM features specific protection schemes, typically for resonant converters when it is necessary to shut down the PWM outputs in a delayed manner, either once the active pulse is completed or once a push-pull period is completed. These features are enabled with DLYPRTEN bit in the HRTIM_OUTxR register, and are using specific external event channels.

Delayed idle

In this mode, the active pulse is completed before the protection is activated. The selected external event causes the output to enter in idle mode at the end of the active pulse (defined by an output reset event in HRTIM_RSTx1R or HRTIM_RSTx2R).

Once the protection is triggered, the idle mode is permanently maintained but the counter continues to run, until the output is re-enabled. Tx1OEN and Tx2OEN bits are not affected by the delayed idle entry. To exit from delayed idle and resume operation, it is necessary to overwrite Tx1OEN and Tx2OEN bits to 1. The output state changes on the first transition to an active state following the output enable command.

Note: The delayed idle mode cannot be exited immediately after having been entered, before the active pulse is completed: it is mandatory to make sure that the outputs are in idle state

before resuming the run mode. This can be done by waiting up to the next period, for instance, or by polling the O1CPY and/or O2CPY status bits in the TIMxISR register.

The delayed idle mode can be applied to a single output (DLYPRT[2:0] = x00 or x01) or to both outputs (DLYPRT[2:0] = x10).

An interrupt or a DMA request can be generated in response to a Delayed Idle mode entry. The DLYPRT flag in HRTIM_TIMxISR is set as soon as the external event arrives, independently from the end of the active pulse on output.

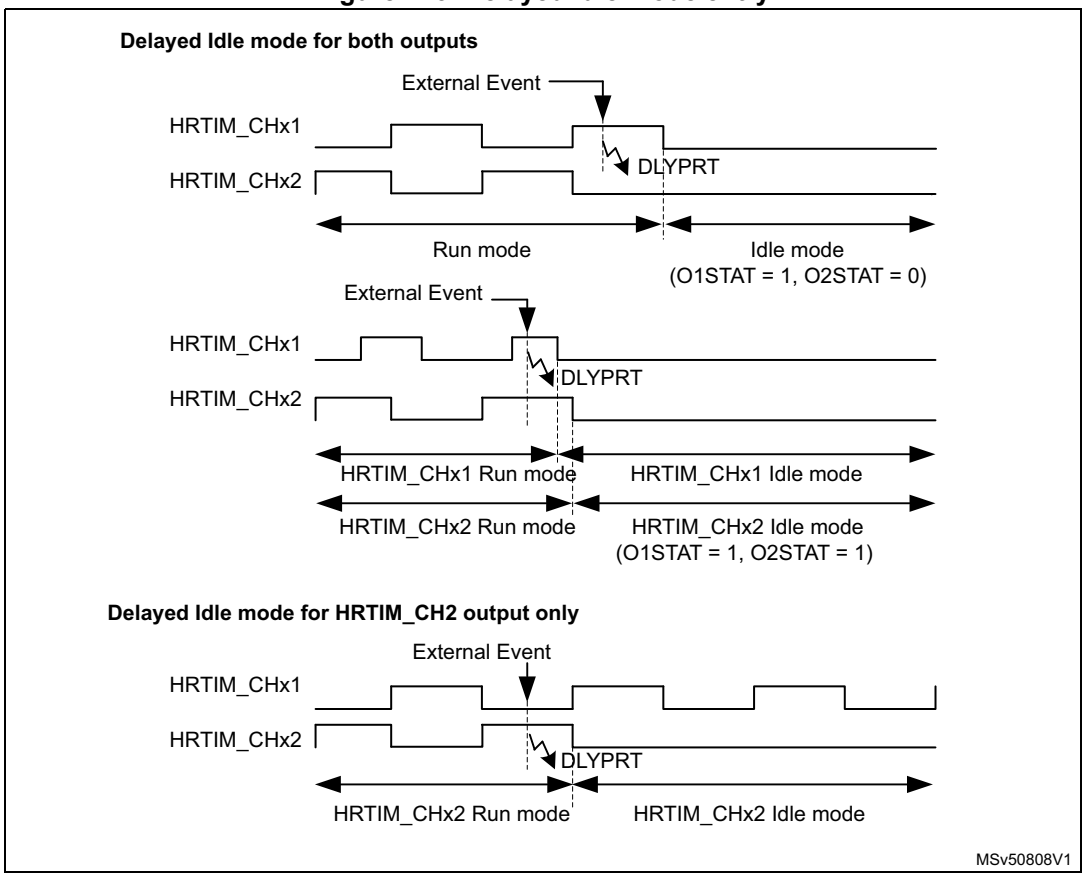
When the Delayed Idle mode is triggered, the output states can be determined using O1STAT and O2STAT in HRTIM_TIMxISR. Both status bits are updated even if the delayed idle is applied to a single output. When the push-pull mode is enabled, the IPPSTAT flag in HRTIM_TIMxISR indicates during which period the delayed protection request occurred.

This mode is available whatever the timer operating mode (regular, push-pull, deadtime). It is available with some external events only:

- EEVF and EEVG for timer A, B and C
- EEVH and EEVI for timer D, E and F

The delayed protection mode can be triggered only when the counter is enabled (TxCEN bit set). It remains active even if the TxEN bit is reset, until the TxyOEN bits are set.

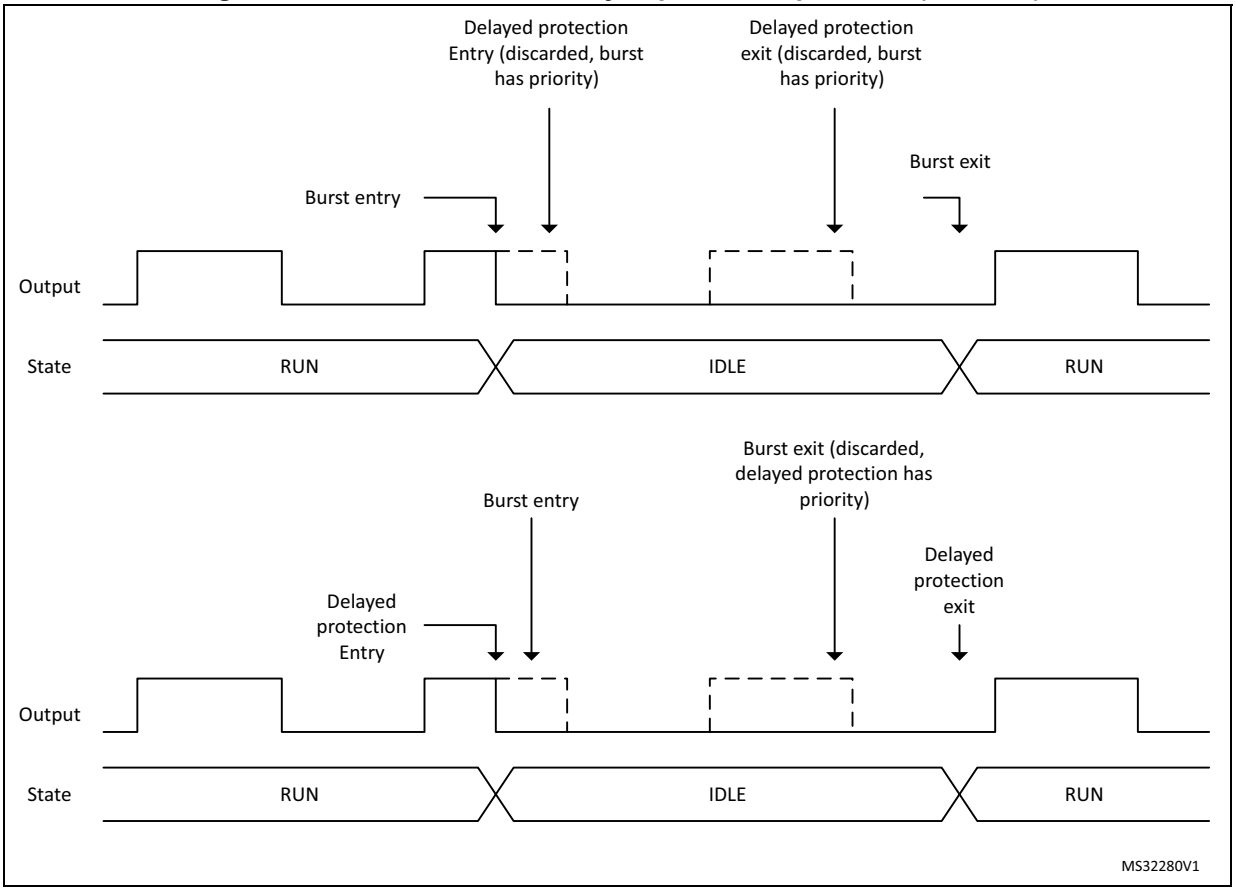
Figure 473. Delayed Idle mode entry



The delayed idle mode has a higher priority than the burst mode: any burst mode exit request is discarded once the delayed idle protection has been triggered. On the contrary, If the delayed protection is exited while the burst mode is active, the burst mode is resumed

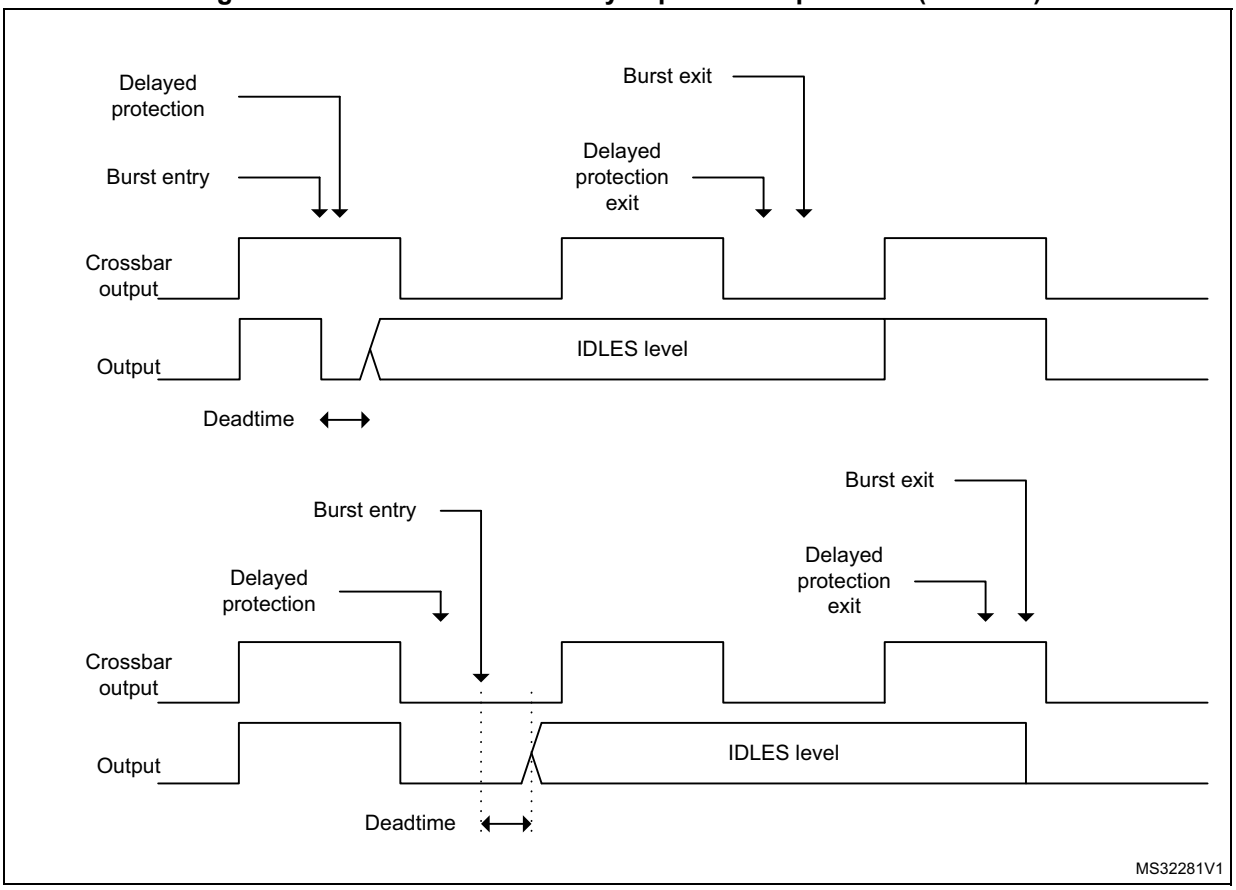
normally and the output is maintained in the idle state until the burst mode exits. *Figure 474* gives an overview of these different scenarios.

Figure 474. Burst mode and delayed protection priorities (DIDL = 0)



The same priorities are applied when the delayed burst mode entry is enabled (DIDL bit set), as shown on *Figure 475* below.

Figure 475. Burst mode and delayed protection priorities (DIDL = 1)



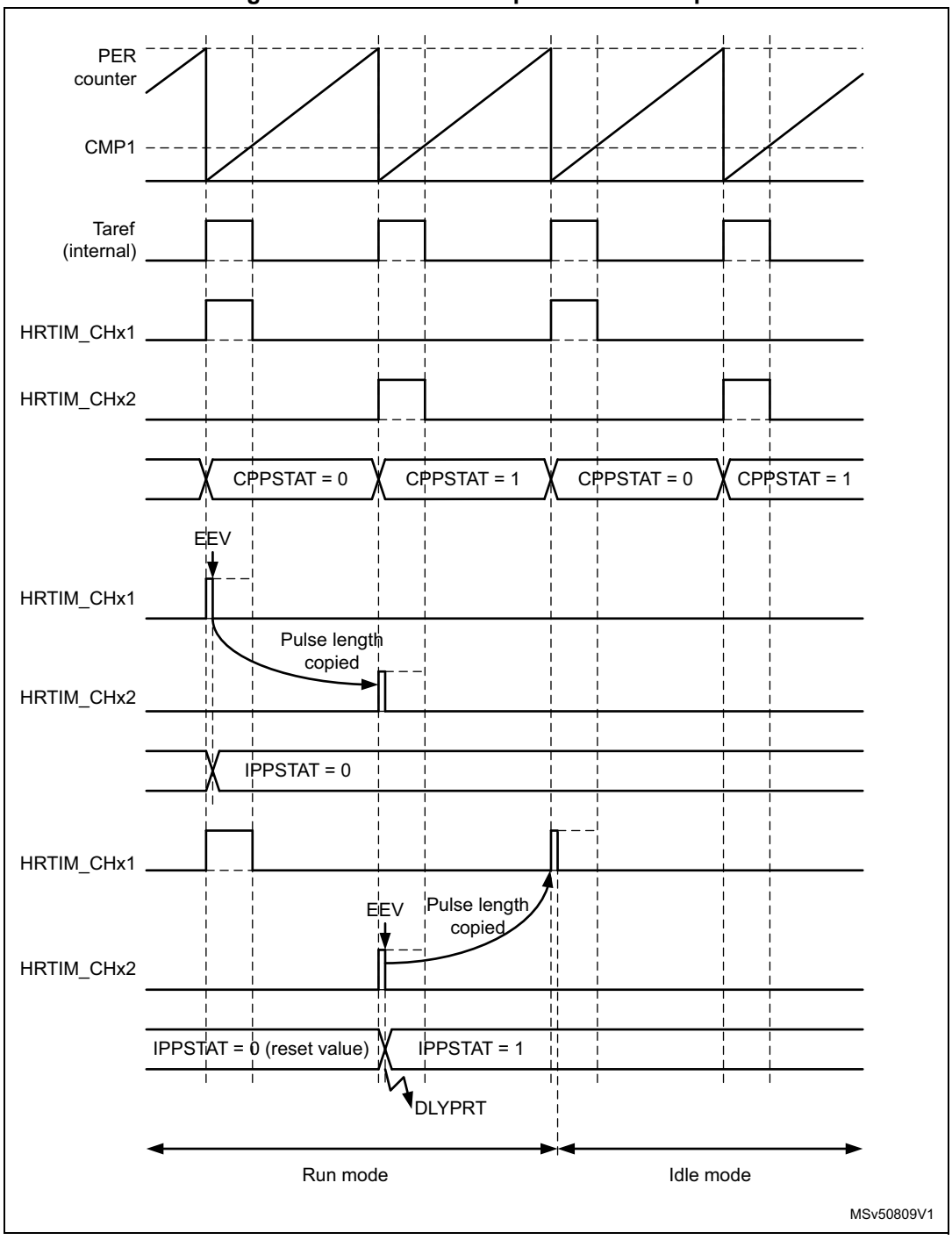
Balanced idle

Only available in push-pull mode, the balanced idle allows to have a balanced pulsewidth on the two outputs when one of the active pulse is shortened due to a protection. The pulsewidth, terminated earlier than programmed, is copied on the alternate output, then the two outputs are put in idle state, until the normal operation is resumed by software. This mode is enabled by writing x11 in DLYPRT[2:0] bitfield in HRTIM_OUTxR.

This mode is available with only 2 external events:

- EEVF and EEVG for timer A, B and C
- EEVH and EEVI for timer D, E and F

Figure 476. Balanced Idle protection example



When the balanced Idle mode is enabled, the selected external event triggers a capture of the counter value into the compare 4 active register (this value is not user-accessible). The push-pull is maintained for one additional period so that the shorten pulse can be repeated: a new output reset event is generated while the regular output set event is maintained.

The Idle mode is then entered and the output takes the level defined by IDLESx bits in the HRTIM_OUTxR register. The balanced idle mode entry is indicated by the DLYPRT flag,

while the IPPSTAT flag indicates during which period the external event occurred, to determine the sequence of shorten pulses (A1 then A2 or vice versa).

The timer operation is not interrupted (the counter continues to run).

To enable the balanced idle mode, it is necessary to have the following initialization:

- Timer operating in continuous mode (CONT = 1)
- Push-pull mode enabled
- HRTIM_CMP4xR must be set to 0 and the content transferred into the active register (for instance by forcing a software update)
- DELCMP4[1:0] bit field must be set to 00 (autodelayed mode disabled)
- DLYPRT[2:0] = x11 (delayed protection enable)

Note: The HRTIM_CMP4xR register must not be written during a balanced idle operation. The CMP4 event is reserved and cannot be used for another purpose.

In balanced idle mode, it is recommended to avoid multiple external events or software-based reset events causing an output reset. If such an event arrives before a balanced idle request within the same period, it causes the output pulses to be unbalanced (1st pulse length defined by the external event or software reset, while the 2nd pulse is defined by the balanced idle mode entry). The balanced idle mode is applicable only when a single pulse is generated per counting period, as on Figure 476: Balanced Idle protection example.

The minimum pulsewidth that can be handled in balanced idle mode is 4 f_{HRTIM} clock periods.

If the capture occurs before the counter has reached this minimum value, the current pulse is extended up to 4 f_{HRTIM} clock periods before being copied into the secondary output. In any case, the pulsewidths are always balanced.

Tx1OEN and Tx2OEN bits are not affected by the balanced idle entry. To exit from balanced idle and resume the operation, it is necessary to overwrite Tx1OEN and Tx2OEN bits to 1 simultaneously. The output state changes on the first active transition following the output enable.

It is possible to resume operation similarly to the delayed idle entry. For instance, if the external event arrives while output 1 is active (delayed idle effective after output 2 pulse), the re-start sequence can be initiated for output 1 first. To do so, it is necessary to poll CPPSTAT bit in the HRTIM_TIMxISR register. Using the above example (IPPSTAT flag equal to 0), the operation is resumed when CPPSTAT bit is 0.

In order to have a specific re-start sequence, it is possible to poll the CPPSTAT to know which output is active first. This allows, for instance, to re-start with the same sequence as the idle entry sequence: if the external event arrives during output 1 active, the re-start sequence is initiated when the output 1 is active (CPPSTAT = 0).

Note: The balanced idle mode must not be disabled while a pulse balancing sequence is on-going. It is necessary to wait until the CMP4 flag is set, thus indicating that the sequence is completed, to reset the DLYPRTEN bit.

The balanced idle protection mode can be triggered only when the counter is enabled (TxCEN bit set). It remains active even if the TxCEN bit is reset, until TxyOEN bits are set.

Balanced idle can be used together with the burst mode under the following conditions:

- TxBM bit must be reset (counter clock maintained during the burst, see *Section 33.3.15*),
- No balanced idle protection must be triggered while the outputs are in a burst idle state.

The balanced idle mode has a higher priority than the burst mode: any burst mode exit request is discarded once the balanced idle protection has been triggered. On the contrary, if the delayed protection is exited while the burst mode is active, the burst mode is resumed normally.

Note: Although the output state is frozen in idle mode, a number of events are still generated on the auxiliary outputs (see Section 33.3.18) during the idle period following the delayed protection:
- Output set/reset interrupt or DMA requests
- External event filtering based on output signal
- Capture events triggered by set/reset

Balanced idle automatic resuming

The balanced Idle mode can be configured to have an automatic resuming of operation after a trigger.

Once the shorten pulse has been copied to the alternate output, the pulse width is reset to its original value and the timer resumes operation: the two outputs keep on being in RUN mode.

This is enabled by setting the BIAR bit in the HRTIM_OUTxR register.

This mode must be used only when the period in HRTIM_PERxR is greater than 6 periods of the f_{HRTIM} clock.

Note: This bit is only significant if DLYPRT[2:0] = 011 or 111, it is ignored otherwise.

Note: In balanced idle automatic resuming mode, it is mandatory to set the IDLES state to inactive.

33.3.11 Register preload and update management

Most of HRTIM registers are buffered and can be preloaded if needed. Typically, this allows to prevent the waveforms from being altered by a register update not synchronized with the active events (set/reset).

When the preload mode is enabled, accessed registers are shadow registers. Their content is transferred into the active register after an update request, either software or synchronized with an event.

By default, PREEN bits in HRTIM_MCR and HRTIM_TIMxCR registers are reset and the registers are not preloaded: any write directly updates the active registers. If PREEN bit is reset while the timer is running and preload was enabled, the content of the preload registers is directly transferred into the active registers.

Each timing unit and the master timer have their own PREEN bit. If PRREN is set, the preload registers are enabled and transferred to the active register only upon an update event.

There are two options to initialize the timer when the preload feature is needed:

- Enable PREEN bit at the very end of the timer initialization to have the preload registers transferred into the active registers before the timer is enabled (by setting MCEN and TxCEN bits).
- Enable PREEN bit at any time during the initialization and force a software update immediately before starting.

Table 350 lists the registers which can be preloaded, together with a summary of available update events.

Table 350. HRTIM preloadable control registers and associated update sources

Timer	Preloadable registers	Preload enable	Update sources
Master timer	HRTIM_DIER HRTIM_MPER HRTIM_MREP HRTIM_MCMP1R HRTIM_MCMP2R HRTIM_MCMP3R HRTIM_MCMP4R	PREEN bit in HRTIM_MCR	Software Repetition event Burst DMA event Repetition event following a burst DMA event
Timer x x = A..F	HRTIM_TIMxDIER HRTIM_TIMxPER HRTIM_TIMxREP HRTIM_TIMxCMP1R HRTIM_TIMxCMP1CR HRTIM_TIMxCMP2R HRTIM_TIMxCMP3R HRTIM_TIMxCMP4R HRTIM_DTxR HRTIM_SETx1R HRTIM_RSTx1R HRTIM_SETx2R HRTIM_RSTx2R HRTIM_RSTxR	PREEN bit in HRTIM_TIMxCR	Software TIMx repetition event TIMx reset event Burst DMA event Update event from other timers (TIMy, master) Update event following a burst DMA event Update enable inputs hrtim_upd_en[3:1] Update event following an update enable input following an update event on hrtim_upd_en[3:1] inputs
HRTIM Common	HRTIM_ADC1R HRTIM_ADC2R HRTIM_ADC3R HRTIM_ADC4R	TIMx or master timer Update, depending on ADxUSRC[2:0] bits in HRTIM_CR1, if PREEN = 1 in the selected timer	

The master timer has 4 update options:

1. Software: writing 1 into MSWU bit in HRTIM_CR2 forces an immediate update of the registers. In this case, any pending hardware update request is cancelled.
2. Update done when the master counter rolls over and the master repetition counter is equal to 0. This is enabled when MREPU bit is set in HRTIM_MCR.
3. Update done once burst DMA is completed (see *Section 33.3.23* for details). This is enabled when BRSTDMA[1:0] = 01 in HRTIM_MCR. It is possible to have both MREPU=1 and BRSTDMA=01.
   Note: The update can take place immediately after the end of the burst sequence if SWU bit is set (i.e. forced update mode). If SWU bit is reset, the update is done on the next update event following the end of the burst sequence.
4. Update done when the master counter rolls over following a burst DMA completion. This is enabled when BRSTDMA[1:0] = 10 in HRTIM_MCR.

An interrupt or a DMA request can be generated by the master update event.

Each timer (TIMA..F) can also have the update done as follows:

- By software: writing 1 into TxSWU bit in HRTIM_CR2 forces an immediate update of the registers. In this case, any pending hardware update request is canceled.
- Update done when the counter rolls over and the repetition counter is equal to 0. This is enabled when TxREPU bit is set in HRTIM_TIMxCR.
- Update done when the counter is reset or rolls over in continuous mode. This is enabled when TxRSTU bit is set in HRTIM_TIMxCR. This is used for a timer operating in single-shot mode, for instance.
- Update done once a burst DMA is completed. This is enabled when UPDGAT[3:0] = 0001 in HRTIM_TIMxCR.
- Update done on the update event following a burst DMA completion (the event can be enabled with TxRSTU, TxREPU, MSTU or TxU). This is enabled when UPDGAT[3:0] = 0010 in HRTIM_TIMxCR.
- Update done when receiving a request on hrtim_upd_en[3:1]. This is enabled when UPDGAT[3:0] = 0011, 0100, 0101 in HRTIM_TIMxCR.
- Update done on the update event following a request on hrtim_upd_en[3:1] (the event can be enabled with TxRSTU, TxREPU, MSTU or TxU). This is enabled when UPDGAT[3:0] = 0110, 0111, 1000 in HRTIM_TIMxCR
- Update done synchronously with any other timer or master update (for instance TIMA can be updated simultaneously with TIMB). This is used for converters requiring several timers, and is enabled by setting bits MSTU and TxU in HRTIM_TIMxCR register.

The update enable inputs hrtim_upd_en[3:1] allow to have an update event synchronized with on-chip events coming from the general-purpose timers. These inputs are rising-edge sensitive.

Table 327 lists the connections between update enable inputs and the on-chip sources.

This allows to synchronize low frequency update requests with high-frequency signals (for instance an update on the counter roll-over of a 100 kHz PWM that has to be done at a 100 Hz rate).

Note: The update events are synchronized to the prescaler clock when CKPSC[2:0] > 5.

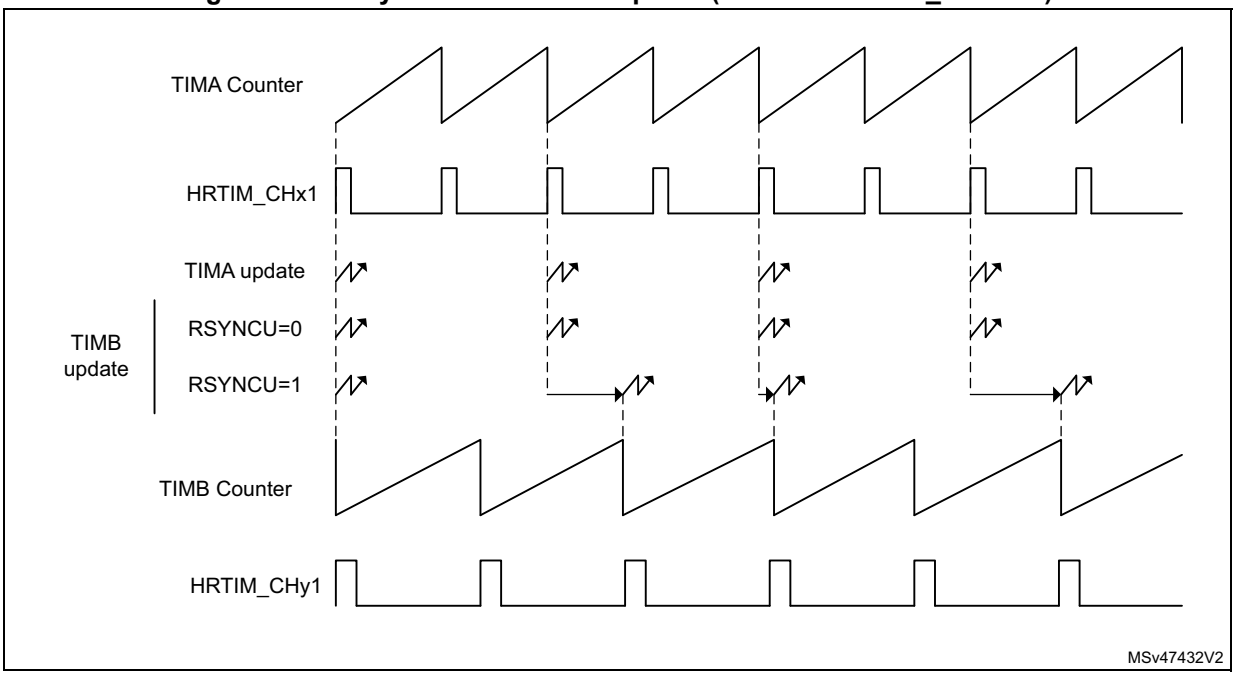
The update coming from adjacent timers (when MSTU, TAU, TBU, TCU, TDU, TEU, TFU bit is set) or from a software update (TxSWU bit) can either be taken into account immediately or re-synchronized with the timers reset/roll-over event. This is done with the RSYNCU bit in the HRTIM_TIMxCR register, as show on *Figure 477* below):

- RSYNCU = 0: The update coming from adjacent timers is taken into account immediately
- RSYNCU = 1: The update coming from adjacent timers is taken into account on the following reset/roll-over event.

The RSYNCU bit is significant only when UPDGAT[3:0] = 0000, it is ignored otherwise.

An interrupt or a DMA request can be generated by the Timx update event.

Figure 477. Resynchronized timer update (TAU=1 in HRTIM_TIMBCR)



MUDIS and TxUDIS bits in the HRTIM_CR1 register allow to temporarily disable the transfer from preload to active registers, whatever the selected update event. This allows to modify several registers in multiple timers. The register update event takes place once these bits are reset, on the next coming update event.

MUDIS and TxUDIS bits are all grouped in the same register. This allows the update of multiple timers (not necessarily synchronized) to be disabled and resumed simultaneously.

The following example is a practical use case. A first power converter is controlled with the master, TIMB and TIMC. TIMB and TIMC must be updated simultaneously with the master timer repetition event. A second converter works in parallel with TIMA, TIMD and TIME, and TIMD, TIME must be updated with TIMA repetition event.

First converter

In HRTIM_MCR, MREPU bit is set: the update occurs at the end of the master timer counter repetition period. In HRTIM_TIMBCR and HRTIM_TIMCCR, MSTU bits are set to have TIMB and TIMC timers updated simultaneously with the master timer.

When the power converter set-point has to be adjusted by software, MUDIS, TBUDIS and TCUDIS bits of the HRTIM_CR register must be set prior to write accessing registers to update the values (for instance the compare values). From this time on, any hardware update request is ignored and the preload registers can be accessed without any risk to have them transferred into the active registers. Once the software processing is over, MUDIS, TBUDIS and TCUDIS bits must be reset. The transfer from preload to active registers is done as soon as the master repetition event occurs.

Second converter

In HRTIM_TIMACR, TAREPU bit is set: the update occurs at the end of the timer A counter repetition period. In HRTIM_TIMDCR and HRTIM_TIMECR, TAU bits are set to have TIMD and TIME timers updated simultaneously with timer A.

When the power converter set-point has to be adjusted by software, TAUDIS, TDUDIS and TEUDIS bits of the HRTIM_CR register must be set prior to write accessing the registers to

update the values (for instance the compare values). From this time on, any hardware update request is ignored and the preload registers can be accessed without any risk to have them transferred into the active registers. Once the software processing is over, TAUDIS, TDUDIS and TEUDIS bits can be reset: the transfer from preload to active registers is done as soon as the timer A repetition event occurs.

Remote update disable

It is possible to disable the update on another on-chip HRTIMER. Refer to *Section 33.3.19: Synchronizing the HRTIM with other timers or HRTIM instances* for details.

33.3.12 PWM mode with “greater than” comparison

A specific no-latency update mode is available for PWM signals generated with the CMP1 and CMP3 registers. It allows to have a new duty cycle value applied as soon as possible within the PWM cycle, without having to wait the completion of the current PWM period. This reduces the overall delay time in software control loops. As shown on *Figure 478* below, this eventually allows to have:

- An early turn-off of the output if the new compare value is below the current counter value and the current compare value is above the counter, at the time the new value is written.
- An early turn-on of the output, re-enabling the output if the new compare value is above the counter value and the current compare value is below the counter, at the time the new value is written.

The output signal is left unchanged when the new compare value and current compare value are both below the counter.

This feature is only available for CMP1 or CMP3 RESET events, and is enabled using the GTCMP1 and GTCMP3 enable bits in the HRTIM_TIMxCR2 register.

The preload mechanism is inactive for a compare register when the corresponding GTCMPx bit is set, whatever the PREEN bit value. This mode is intended to have the new compare value taken into account as soon as possible after a new value write, without waiting for the preload to active register transfer.

These bits are defining the compare 1 and compare 3 operating modes as following

- GTCMPx = 0: the compare x event is generated when the counter is equal to the compare value (compare match mode). If the compare value is changed on-the-fly, the compare event may not be generated.
- GTCMPx = 1: the compare x event is generated when the counter is greater than the compare value. If the compare value is changed on-the-fly, the new compare value is compared with the current counter value and an output SET or RESET can be generated.

The “greater than” compare mode causes the crossbar to act differently depending on comparison result. Let’s consider the CMP1 event is doing an output RESET. When the new compare value is written, two cases are considered

- If the new compare value is below the counter value, the RESET event is issued and can eventually cause an early turn-OFF
- If the new compare value is above the counter value, a SET event is generated so as to re-arm the output value before it is actually RESET when the counter exceeds the counter value (early turn-ON).

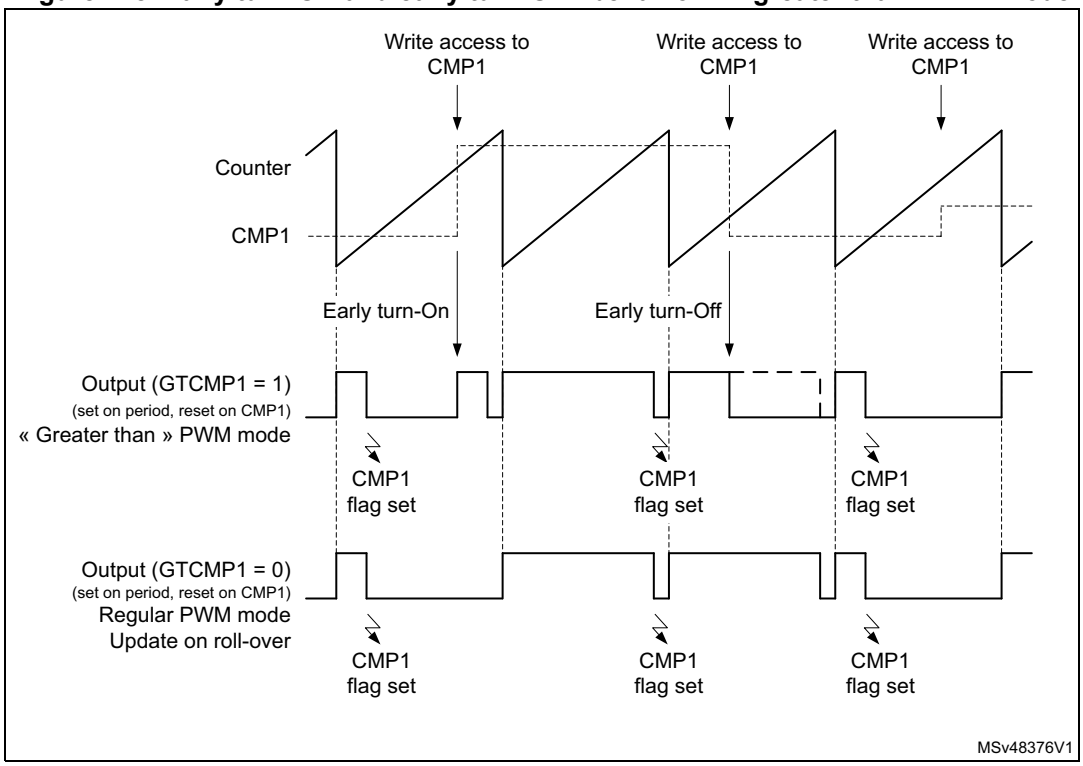
The “greater than” compare mode is supported for both SET and RESET actions.

The “greater than” compare mode must only be used for the following configuration:

1. In the fixed frequency configuration, the period event must trigger the output set and the “greater than” compare triggers the output reset (or vice versa the period must trigger the reset if the “greater-than” compare triggers the set).
2. For variable frequency configuration, the event selected as counter reset source must also be selected as set or reset source for the timer output (opposite direction as the “greater than” compare event).

Note: The “greater-than” modes must not be used when the CMP1 and/or CMP3 modes are controlled by hardware in half and interleaved modes.

Figure 478. Early turn-ON and early turn-OFF behavior in “greater than” PWM mode



The immediate update mode implies that the content of the preload register is transferred into the active register at the very same time the register is written. When GTCMP1 and/or GTCMP3 bits are set, their respective preload mechanism is disabled (for HRTIM_TIMxCMP1 and/or HRTIM_TIMxCMP3 registers), whatever the PREEN bit value.

Note: The compare interrupt flags (CMP1 and CMP3 in HRTIM_TIMxISR) are not generated in case of late turn-ON and early turn-OFF, as shown on Figure 478.

Note: The “Greater than” comparison must not be done on both CMP1 and CMP3 for the same output (GTCMP1 and GTCMP3 bits must not be set simultaneously).

33.3.13 Events propagation within or across multiple timers

The HRTIM offers many possibilities for cascading events or sharing them across multiple timing units, including the master timer, to get full benefits from its modular architecture. These are key features for converters requiring multiple synchronized outputs.

This section summarizes the various options and specifies whether and how an event is propagated within the HRTIM.

TIMx update triggered by the master timer update

The sources listed in *Table 351* are generating a master timer update. The table indicates if the source event can be used to trigger a simultaneous update in any of TIMx timing units.

Operating condition: MSTU bit is set in HRTIM_TIMxCR register.

Table 351. Master timer update event propagation

Source	Condition	Propagation	Comment
Burst DMA end	BRSTDMA[1:0] = 01	No	Must be done in TIMxCR (UPDGAT[3:0] = 0001)
Roll-over event following a burst DMA end	BRSTDMA[1:0] = 10	Yes	-
Repetition event caused by a counter roll-over	MREPU = 1	Yes	-
Repetition event caused by a counter reset (from HRTIM_SCIN or software)		No	-
Software update	MSWU = 1	No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update

TIMx update triggered by the TIMy update

The sources listed in *Table 352* are generating a TIMy update. The table indicates if the given event can be used to trigger a simultaneous update in another or multiple TIMx timers.

Operating condition: TyU bit set in HRTIM_TIMxCR register (source = TIMy and destination = TIMx).

Table 352. TIMx update event propagation

Source	Condition	Propagation	Comment
Burst DMA end	UPDGAT[3:0] = 0001	No	Must be done directly in HRTIM_TIMxCR (UPDGAT[3:0] = 0001)
Update caused by the update enable inputs hrtim_upd_en[3:1]	UPDGAT[3:0] = 0011, 0100, 0101	No	Must be done directly in HRTIM_TIMxCR (UPDGAT[3:0] = 0011, 0100, 0101
Master update	MSTU = 1 in HRTIM_TIMyCR	No	Must be done with MSTU = 1 in HRTIM_TIMxCR
Another TIMx update (TIMz>TIMy>TIMx)	TzU=1 in HRTIM_TIMyCR TyU=1 in TIMxCR	No	Must be done with TzU=1 in HRTIM_TIMxCR TzU=1 in HRTIM_TIMyCR
Repetition event caused by a counter roll-over	TyREPU = 1	Yes	-

Table 352. TIMx update event propagation (continued)

Source	Condition	Propagation	Comment
Repetition event caused by a counter reset	TyREPU = 1	-	Refer to counter reset cases below
Counter roll-over	TyRSTU = 1	Yes	-
Counter software reset	TyRST=1 in HRTIM_CR2	No	Can be done simultaneously with update in HRTIM_CR2 register
Counter reset caused by a TIMz compare	TIMzCMPn in HRTIM_RSTyR	Yes	-
Counter reset caused by external events	EXTEVNTn in HRTIM_RSTyR	Yes	-
Counter reset caused by a master compare or a master period	MSTCMPn or MSTPER in HRTIM_RSTyR	Yes	-
Counter reset caused by a TIMy compare	CMPn in HRTIM_RSTyR	Yes	-
Counter reset caused by an update	UPDT in HRTIM_RSTyR	No	Propagation would result in a lock-up situation (update causing reset causing update)
Counter reset caused by HRTIM_SCIN	SYNCRSTy in HRTIM_TIMyCR	No	-
Software update	TySWU = 1	No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update

TIMx counter reset causing a TIMx update

Table 353 lists the counter reset sources and indicates whether they can be used to generate an update.

Operating condition: TxRSTU bit in HRTIM_TIMxCR register.

Table 353. Reset events able to generate an update

Source	Condition	Propagation	Comment
Counter roll-over	-	Yes	-
Update event	UPDT in HRTIM_RSTxR	No	Propagation would result in a lock-up situation (update causing a reset causing an update)
External event	EXTEVNTn in HRTIM_RSTxR	Yes	-
TIMy compare	TIMyCMPn in HRTIM_RSTxR	Yes	-
Master compare	MSTCMPn in HRTIM_RSTxR	Yes	-
Master period	MSTPER in HRTIM_RSTxR	Yes	-
Compare 2 and 4	CMPn in HRTIM_RSTxR	Yes	-
Software	TxRST=1 in HRTIM_CR2	Yes	-
HRTIM_SCIN	SYNCRSTx in HRTIM_TIMxCR	Yes	-

TIMx update causing a TIMx counter reset

Table 354 lists the update event sources and indicates whether they can be used to generate a counter reset.

Operating condition: UPDT bit set in HRTIM_RSTxR.

Table 354. Update event propagation for a timer reset

Source	Condition	Propagation	Comment
Burst DMA end	UPDGAT[3:0] = 0001	Yes	-
Update caused by the update enable inputs hrtim_upd_en[3:1]	UPDGAT[3:0] = 0011, 0100, 0101	Yes	-
Master update caused by a roll-over after a burst DMA	MSTU = 1 in HRTIM_TIMxCR BRSTDMA[1:0] = 10 in HRTIM_MCR	Yes	-

Table 354. Update event propagation for a timer reset (continued)

Source	Condition	Propagation	Comment
Master update caused by a repetition event following a roll-over	MSTU = 1 in HRTIM_TIMxCR MREPU = 1 in HRTIM_MCR	Yes	-
Master update caused by a repetition event following a counter reset (software or due to HRTIM_SCIN)		No	-
Software triggered master timer update	MSTU = 1 in HRTIM_TIMxCR MSWU = 1 in HRTIM_CR2	No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update
TIMy update caused by a TIMy counter roll-over	TyU = 1 in HRTIM_TIMxCR TyRSTU = 1 in HRTIM_TIMyCR	Yes	-
TIMy update caused by a TIMy repetition event	TyU = 1 in HRTIM_TIMxCR TyREPU = 1 in HRTIM_TIMyCR	Yes	-
TIMy update caused by an external event or a TIMy compare (through a TIMy reset)	TyU = 1 in HRTIM_TIMxCR TyRSTU = 1 in HRTIM_TIMyCR EXTEVNTn or CMP4/2 in HRTIM_RSTyCR	Yes	-
TIMy update caused by sources other than those listed above	TyU = 1 in HRTIM_TIMxCR	No	-
Repetition event following a roll-over	TxREPU = 1 in HRTIM_TIMxCR	Yes	-
Repetition event following a counter reset		No	-
Timer reset	TxRSTU = 1 in HRTIM_TIMxCR	No	Propagation would result in a lock-up situation (reset causing an update causing a reset)
Software	TxSWU in HRTIM_CR2	No	-

33.3.14 Output management

Each timing unit controls a pair of outputs. The outputs have three operating states:

- RUN: this is the main operating mode, where the output can take the active or inactive level as programmed in the crossbar unit.
- IDLE: this state is the default operating state after an HRTIM reset, when the outputs are disabled by software or during a burst mode operation (where outputs are temporarily disabled during a normal operating mode; refer to *Section 33.3.15* for more details). It is either permanently active or inactive.
- FAULT: this is the safety state, entered in case of a shut-down request on FAULTx inputs. It can be permanently active, inactive, or Hi-Z.

The output status is indicated by TxyOEN bit in HRTIM_OENR register and TxyODS bit in HRTIM_ODSR register, as in *Table 355*.

Table 355. Output state programming, x= A..F, y = 1 or 2

TxyOEN (control/status) (set by software, cleared by hardware)	TxyODS (status)	Output operating state
1	x	RUN
0	0	IDLE
0	1	FAULT

TxyOEN bit is both a control and a status bit: it must be set by software to have the output in RUN mode. It is cleared by hardware when the output goes back in IDLE or FAULT mode. When TxyOEN bit is cleared, TxyODS bit indicates whether the output is in the IDLE or FAULT state. A third bit in the HRTIM_ODISR register is used to disable the output by software.

Figure 479. Output management overview

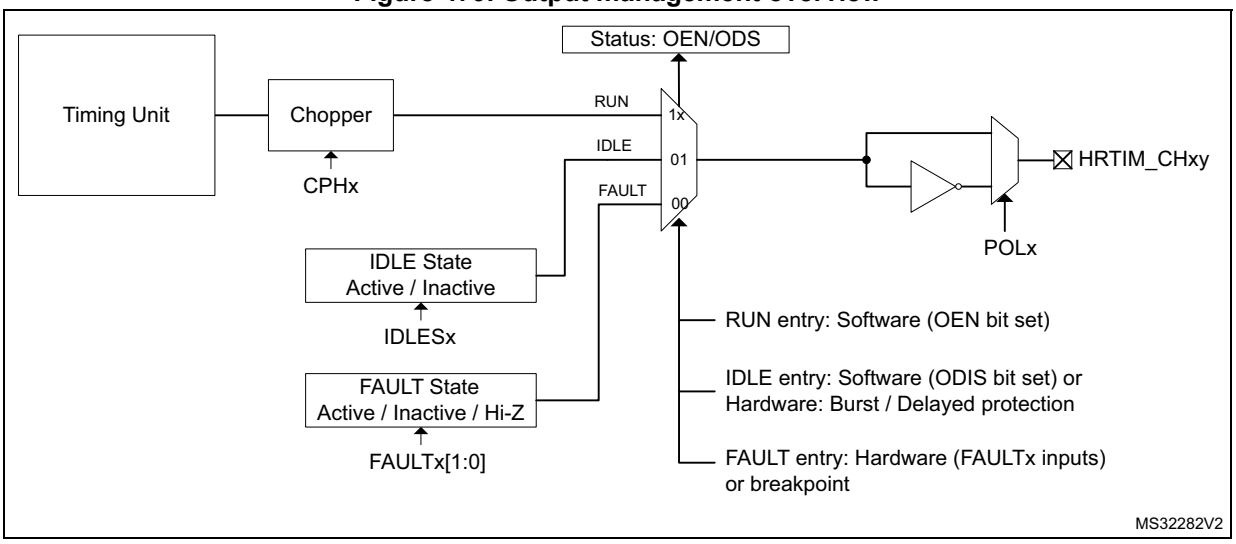
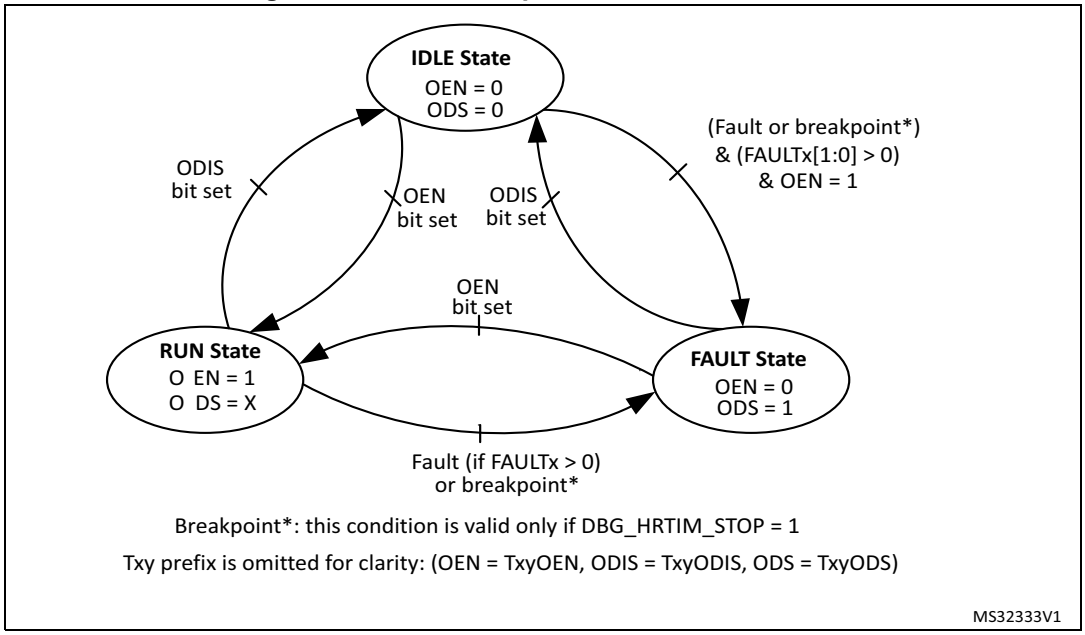


Figure 480 summarizes the bit values for the three states and how the transitions are triggered. Faults can be triggered by any external or internal fault source, as listed in

Section 33.3.17, while the IDLE state can be entered when the burst mode or delayed protections are active.

Figure 480. HRTIM output states and transitions



The FAULT and IDLE levels are defined as active or inactive. Active (or inactive) refers to the level on the timer output that causes a power switch to be closed (or opened for an inactive state).

The IDLE state has the highest priority: the transition FAULT → IDLE is possible even if the FAULT condition is still valid, triggered by ODIS bit set.

The FAULT state has priority over the RUN state: if TxyOEN bit is set simultaneously with a fault event, the FAULT state is entered. The condition is given on the transition IDLE → FAULT, as in *Figure 480*: fault protection needs to be enabled (FAULTx[1:0] bits = 01, 10, 11) and the Txy OEN bit set with a fault active (or during a breakpoint if DBG_HRTIM_STOP = 1).

The output polarity is programmed using POLx bits in HRTIM_OUTxR. When POLx = 0, the polarity is positive (output active high), while it is active low in case of a negative polarity (POLx = 1). Practically, the polarity is defined depending on the power switch to be driven (PMOS versus NMOS) or on a gate driver polarity.

The output level in the FAULT state is configured using FAULTx[1:0] bits in HRTIM_OUTxR, for each output, as follows:

- 00: output never enters the fault state and stays in RUN or IDLE state
- 01: output at active level when in FAULT
- 10: output at inactive level when in FAULT
- 11: output is tri-stated when in FAULT. The safe state must be forced externally with pull-up or pull-down resistors, for instance.

Note: FAULTx[1:0] bits must not be changed as long as the outputs are in FAULT state.

The level of the output in IDLE state is configured using IDLESx bit in HRTIM_OUTxR, as follows:

- 0: output at inactive level when in IDLE
- 1: output at active level when in IDLE

When TxyOEN bit is set to enter the RUN state, the output is immediately connected to the crossbar output. If the timer clock is stopped, the level is either inactive (after an HRTIM reset) or corresponds to the RUN level (when the timer is stopped and the output disabled).

During the HRTIM initialization, the output level can be prepositioned prior to have it in RUN mode, using the software forced output set and reset in the HRTIM_SETx1R and HRTIM_RSTx1R registers.

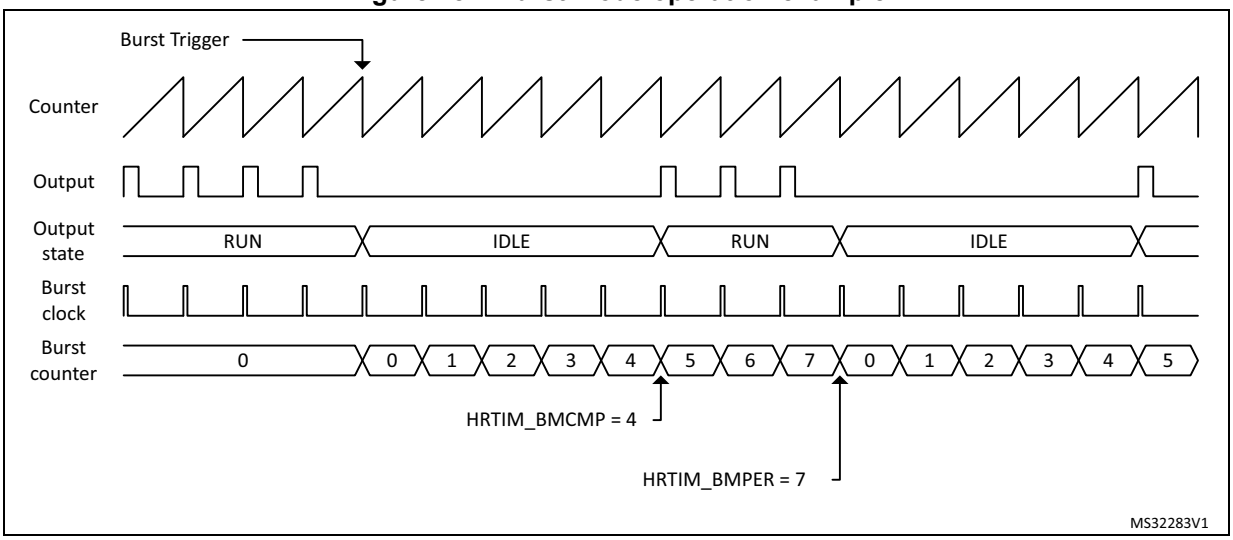
33.3.15 Burst mode controller

The burst mode controller is used to have the outputs alternatively in IDLE and RUN state, by hardware, so as to skip some switching periods with a programmable periodicity and duty cycle.

Burst mode operation is of common use in power converters when operating under light loads. It can significantly increase the efficiency of the converter by reducing the number of transitions on the outputs and the associated switching losses.

When operating in burst mode, one or a few pulses are outputted followed by an idle period equal to several counting periods, typically, where no output pulses are produced, as shown in the example on *Figure 481*.

Figure 481. Burst mode operation example



The burst mode controller consists of:

- A counter that can be clocked by various sources, either within or outside the HRTIM (typically the end of a PWM period).
- A compare register to define the number of idle periods: HRTIM_BMCMP.
- A period register to define the burst repetition rate (corresponding to the sum of the idle and run periods): HRTIM_BMPER.

The burst mode controller is able to take over the control of any of the 10 PWM outputs. The state of each output during a burst mode operation is programmed using IDLESx and IDLEMx bits in the HRTIM_OUTxR register, as in *Table 356*.

Table 356. Timer output programming for burst mode

IDLEMx	IDLESx	Output state during burst mode
0	X	No action: the output is not affected by the burst mode operation.
1	0	Output inactive during the burst
1	1	Output active during the burst

Note: IDLEMx bit must not be changed while the burst mode is active.

The burst mode controller only acts on the output stage. A number of events are still generated during the idle period:

- Output set/reset interrupt or DMA requests
- External event filtering based on Tx2 output signal
- Capture events triggered by output set/reset

During the burst mode, neither start nor reset events are generated on the hrtim_out_sync[2:1] output, even if TxBM bit is set.

Operating mode

It is necessary to have the counter enabled (TxCEN bit set) before using the burst mode on a given timing unit. The burst mode is enabled with BME bit in the HRTIM_BMCR register.

It can operate in continuous or single-shot mode, using BMOM bit in the HRTIM_BMCR register. The continuous mode is enabled when BMOM = 1. The burst operation is maintained until BMSTAT bit in HRTIM_BMCR is reset to terminate it.

In single-shot mode (BMOM = 0), the idle sequence is executed once, following the burst mode trigger, and the normal timer operation is resumed immediately after.

The duration of the idle and run periods is defined with a burst mode counter and two registers. The HRTIM_BMCMPR register defines the number of counts during which the selected timer(s) are in an idle state (idle period). HRTIM_BMPER defines the overall burst mode period (sum of the idle and run periods). Once the initial burst mode trigger has occurred, the idle period length is HRTIM_BMCMPR+1, the overall burst period is HRTIM_BMPER+1.

Note: The burst mode period must not be less than or equal to the deadtime duration defined with DTRx[8:0] and DTFx[8:0] bitfields.

The counters of the timing units and the master timer can be stopped and reset during the burst mode operation. HRTIM_BMCR holds six control bits for this purpose: MTBM (master) and TABM..TEBM for timer A..E.

When MTBM or TxBM bit is reset, the counter clock is maintained. This is used to keep a phase relationship with other timers in multiphase systems, for instance.

When MTBM or TxBM bit is set, the corresponding counter is stopped and maintained in reset state during the burst idle period. This is used to have the timer restarting a full period when exiting from idle. If SYNCSRC[1:0] = 00 or 10 (synchronization output on the master

start or timer A start), a pulse is sent on the HRTIM_SCOUT output when exiting the burst mode.

Note: TxBM bit must not be set when the balanced idle mode is active (DLYPRT[1:0] = 0x11).

Burst mode clock

The burst mode controller counter can be clocked by several sources, selected with BMCLK[3:0] bits in the HRTIM_BMCR register:

- BMCLK[3:0] = 0000 to 0101: master timer and TIMA..E reset/roll-over events. This is used to have burst mode idle and run periods aligned with the timing unit counting period (both in free-running and counter reset mode).
- BMCLK[3:0] = 0110 to 1001: The clocking is provided by the hrtim_bm_ck[4:1] inputs connected to general purpose timers, as in *Table 328*. In this case, the burst mode idle and run periods are not necessarily aligned with timing unit counting period (a pulse on the output may be interrupted, resulting in a waveform with modified duty cycle for instance.
- BMCLK[3:0] = 1010: The f_{HRTIM} clock prescaled by a factor defined with BMPRSC[3:0] bits in HRTIM_BMCR register. In this case, the burst mode idle and run periods are not necessarily aligned with the timing unit counting period (a pulse on the output may be interrupted, resulting in a waveform with a modified duty cycle, for instance.

The pulse width on TIMx OC output must be at least N f_{HRTIM} clock cycles long to be detected by the HRTIM burst mode controller.

Burst mode triggers

To trigger the burst operation, 32 sources are available and are selected using the HRTIM_BMTRGR register:

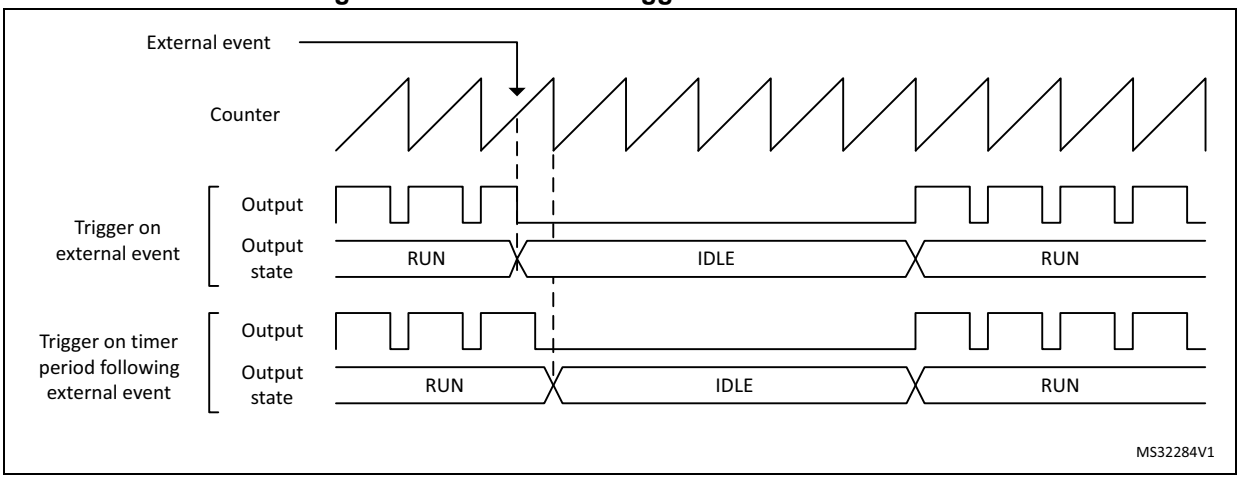
- Software trigger (set by software and reset by hardware)
- 6 master timer events: repetition, reset/roll-over, compare 1 to 4
- 5 x 4 events from timers A..F: repetition, reset/roll-over, compare 1 and 2
- External event G (including TIMA event filtering) and H (including TIMD event filtering)
- Timer A period following external event G (including TIMA event filtering)
- Timer D period following external event H (including TIMD event filtering)
- An on-chip event on the hrtim_bm_trg input (connected to the general-purpose timer TRGO output), see *Table 325* for details.

These sources can be combined to have multiple concurrent triggers.

Burst mode is not retriggerable. In continuous mode, new triggers are ignored until the burst mode is terminated, while in single-shot mode, the triggers are ignored until the current burst completion including run periods (HRTIM_BMPER+1 cycles). This is also valid for software trigger (the software bit is reset by hardware even if it is discarded).

Figure 482 shows how the burst mode is started in response to an external event, either immediately or on the timer period following the event.

Figure 482. Burst mode trigger on external event



For TAEEVG and TDEEVH combined triggers (trigger on a timer period following an external event), the external event detection is always active, regardless of the burst mode programming and the ongoing burst operation:

- When the burst mode is enabled (BME=1) or the trigger is enabled (TAEEVG or TDEEVH bit set in the HRTIM_BMTRGR register) in between the external event and the timer period event, the burst is triggered.
- The single-shot burst mode is retriggered even if the external event occurs before the burst end (as long as the corresponding period happens after the burst).

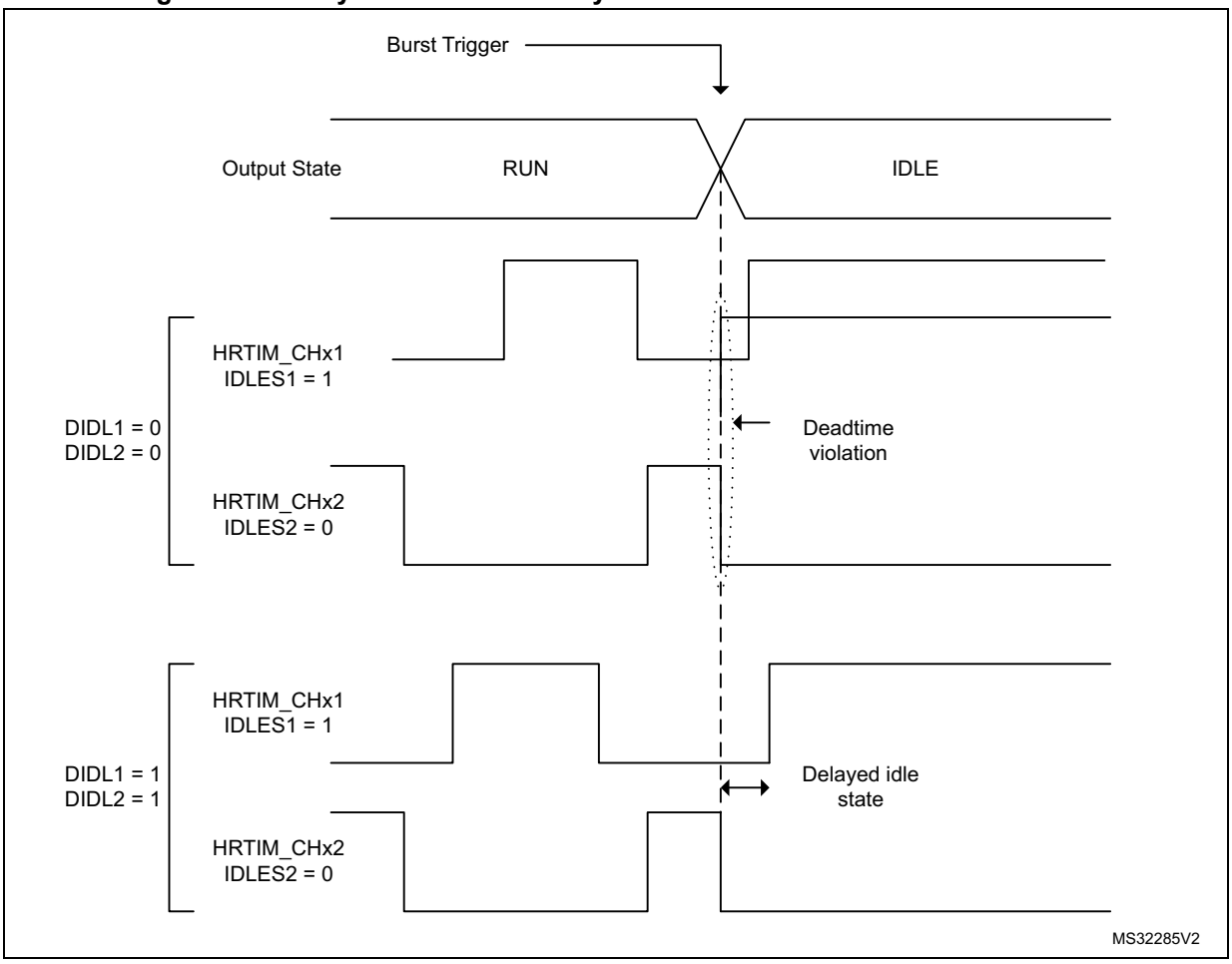
Note: TAEEVG and TDEEVH triggers are valid only after a period event. If the counter is reset before the period event, the pending hrtim_eev7 or hrtim_eev8 event is discarded.

Burst mode delayed entry

By default, the outputs are taking their idle level (as per IDLES1 and IDLES2 setting) immediately after the burst mode trigger.

It is also possible to delay the burst mode entry and force the output to an inactive state during a programmable period before the output takes its idle state. This is useful when driving two complementary outputs, one of them having an active idle state, to avoid a deadtime violation as shown on *Figure 483*. This prevents any risk of shoot through current in half-bridges, but causes a delayed response to the burst mode entry.

Figure 483. Delayed burst mode entry with deadtime enabled and IDLESx = 1



The delayed burst entry mode is enabled with DIDLx bit in the HRTIM_OUTxR register (one enable bit per output). It forces a deadtime insertion before the output takes its idle state. Each TIMx output has its own deadtime value:

- DTRx[8:0] on output 1 when DIDL1 = 1
- DTFx[8:0] on output 2 when DIDL2 = 1

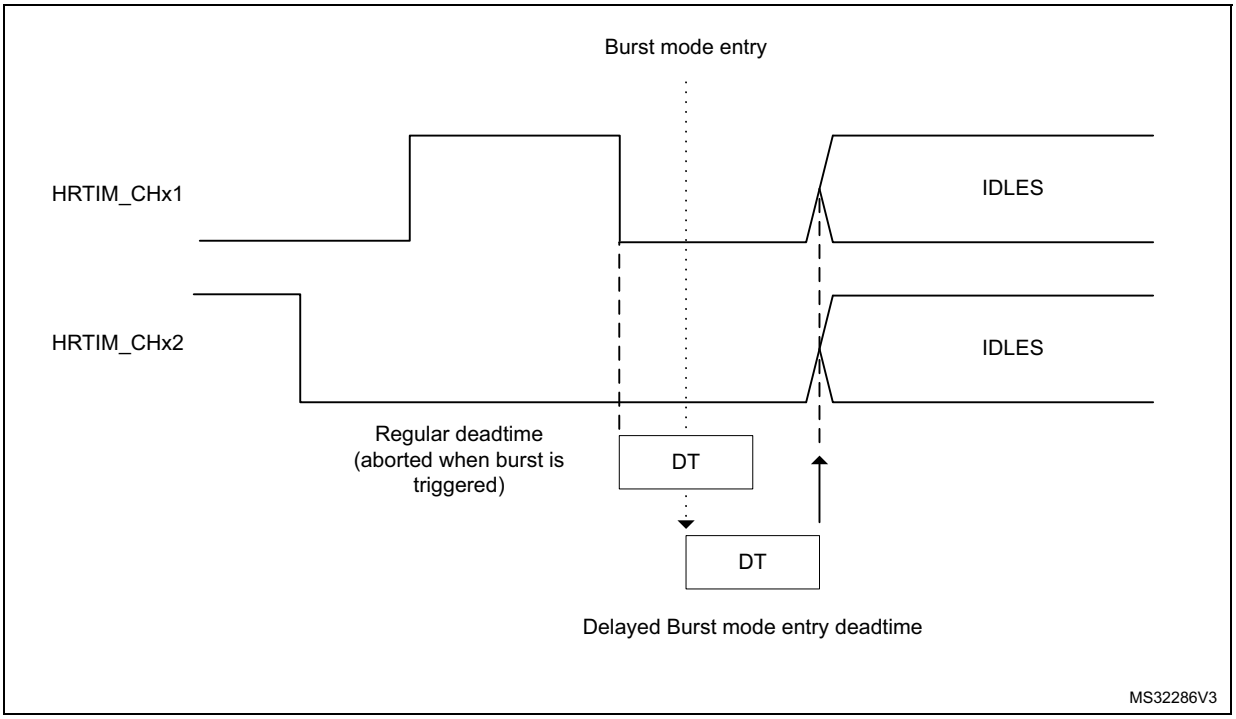
DIDLx bits can be set only if one of the outputs has an active idle level during the burst mode (IDLES = 1) and only when positive deadtimes are used (SDTR/SDTF set to 0).

Note: The delayed burst entry mode uses deadtime generator resources. Consequently, when any of the two DIDLx bits is set and the corresponding timing unit uses the deadtime insertion (DTEN bit set in HRTIM_OUTxR), it is not possible to use the timerx output 2 as a filter for external events (Tx2 filtering signal is not available).

When durations defined by DTRx[8:0] and DTFx[8:0] are lower than three f_{HRTIM} clock cycle periods, the limitations related to the narrow pulse management listed in *Section 33.3.7* must be applied.

When the burst mode entry arrives during the regular deadtime, it is aborted and a new deadtime is restarted corresponding to the inactive period, as on *Figure 484*.

Figure 484. Delayed burst mode entry during deadtime



Burst mode exit

The burst mode exit is either forced by software (in continuous mode) or once the idle period is elapsed (in single-shot mode). In both cases, the counter is re-started immediately (if it was hold in a reset state with MTBM or TxBM bit = 1), but the effective output state transition from the idle to active mode only happens after the programmed set/reset event.

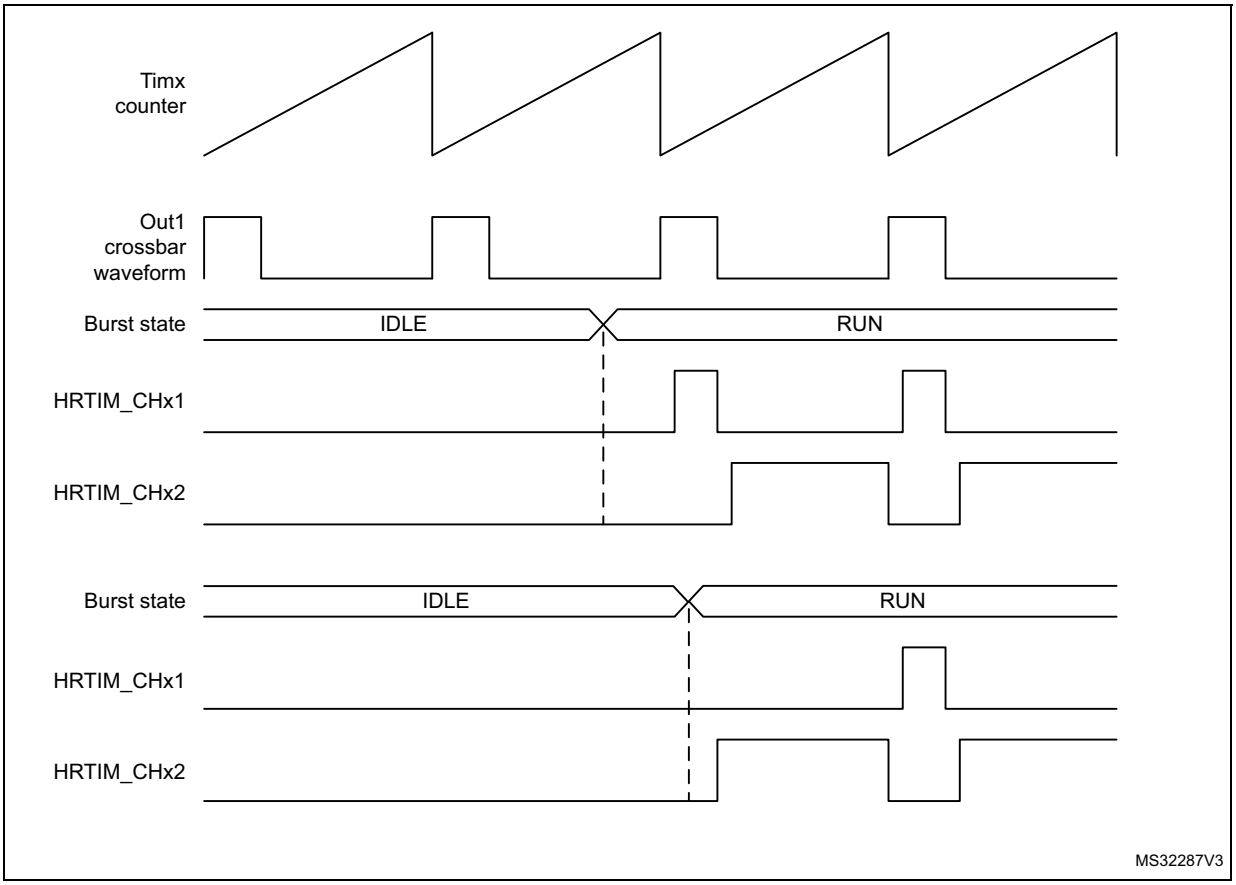
A burst period interrupt is generated in single-shot and continuous modes when BMPERIE enable bit is set in the HRTIM_IER register. This interrupt can be used to synchronize the burst mode exit with a burst period in continuous burst mode.

Figure 485 shows how a normal operation is resumed when the deadtime is enabled. Although the burst mode exit is immediate, this is only effective on the first set event on any of the complementary outputs.

Two different cases are presented:

1. The burst mode ends while the signal is inactive on the crossbar output waveform. The active state is resumed on Tx1 and Tx2 on the set event for the Tx1 output, and the Tx2 output does not take the complementary level on burst exit.
2. The burst mode ends while the crossbar output waveform is active: the activity is resumed on the set event of Tx2 output, and Tx1 does not take the active level immediately on burst exit.

Figure 485. Burst mode exit when the deadtime generator is enabled



The behavior described above is slightly different when the push-pull mode is enabled. The push-pull mode forces an output reset at the beginning of the period if the output is inactive, or symmetrically forces an active level if the output was high during the preceding period.

Consequently, an output with an active idle state can be reset at the time the burst mode is exited even if no transition is explicitly programmed. For symmetrical reasons, an output can be set at the time the burst mode is exited even if no transition is explicitly programmed, in case it was active when it entered in idle state.

Burst mode registers preloading and update

BMPREN bit (burst mode preload enable) is used to have the burst mode compare and period registers preloaded (HRTIM_BMCMP and HRTIM_BMPER).

When BMPREN is set, the transfer from preload to active register happens:

- When the burst mode is enabled (BME = 1),
- At the end of the burst mode period.

A write into the HRTIM_BMPER period register disables the update temporarily, until the HRTIM_BMCMP compare register is written, to ensure the consistency of the two registers when they are modified.

If the compare register only needs to be changed, a single write is necessary. If the period only needs to be changed, it is also necessary to rewrite the compare to have the new values taken into account.

When BMPREN bit is reset, the write access into BMCMPR and BMPER directly updates the active register. In this case, it is necessary to consider when the update is done during the overall burst period, for the two cases below:

a) Compare register update

If the new compare value is above the current burst mode counter value, the new compare is taken into account in the current period.

If the new compare value is below the current burst mode counter value, the new compare is taken into account in the next burst period in continuous mode, and ignored in single-shot mode (no compare match occurs and the idle state lasts until the end of the idle period).

b) Period register update

If the new period value is above the current burst mode counter value, the change is taken into account in the current period.

Note: If the new period value is below the current burst mode counter value, the new period is not taken into account, the burst mode counter overflows (at 0xFFFF) and the change is effective in the next period. In single-shot mode, the counter rolls over at 0xFFFF and the burst mode restarts for another period up to the new programmed value.

Burst mode emulation using a compound register

The burst mode controller only controls one or a set of timers for a single converter. When the burst mode is necessary for multiple independent timers, it is possible to emulate a simple burst mode controller using the DMA and the HRTIM_CMP1CxR compound register, which holds aliases of both the repetition and the compare 1 registers.

This is applicable to a converter which only requires a simple PWM (typically a buck converter), where the duty cycle only needs to be updated. In this case, the CMP1 register is used to reset the output (and define the duty cycle), while it is set on the period event.

In this case, a single 32-bit write access in CMP1CxR is sufficient to define the duty cycle (with the CMP1 value) and the number of periods during which this duty cycle is maintained (with the repetition value). To implement a burst mode, it is then only necessary to transfer by DMA (upon repetition event) two 32-bit data in continuous mode, organized as follows:

CMPC1xR = {REP_Run; CMP1 = Duty_Cycle}, {REP_Idle; CMP1 = 0}

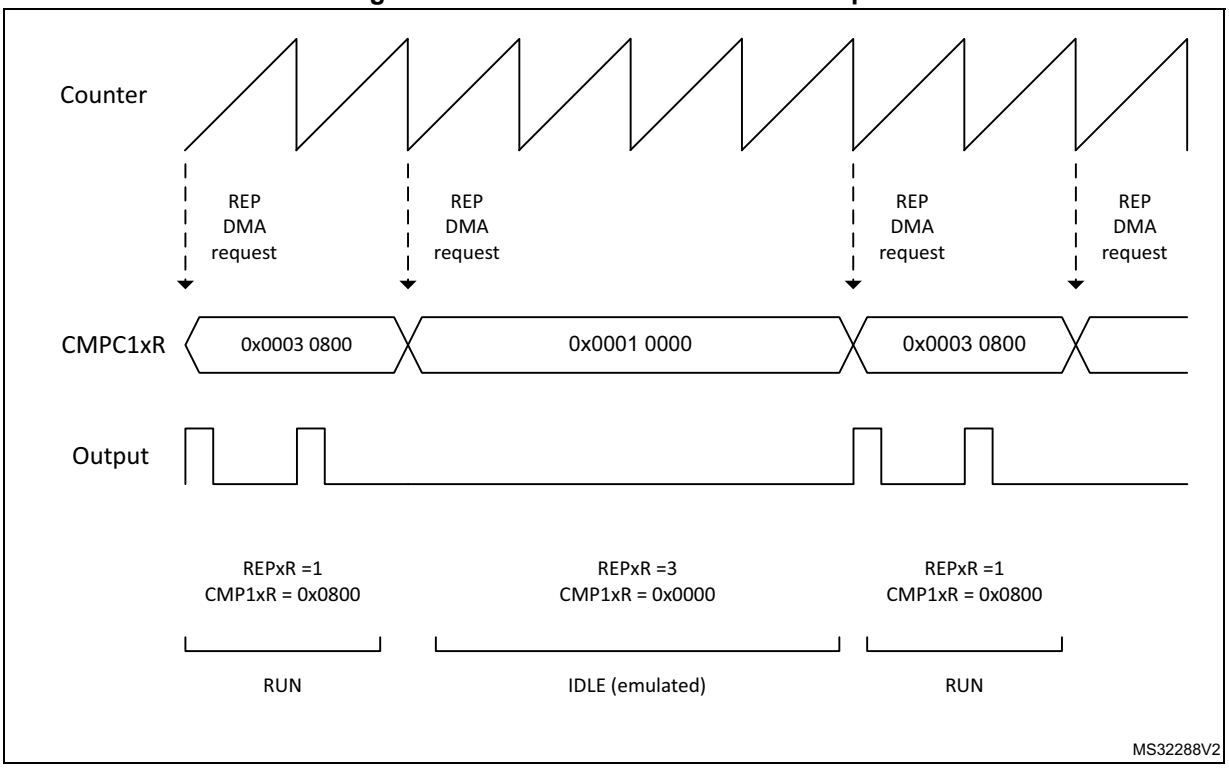
For instance, the values:

{0x0003 0000}: CMP1 = 0 for four periods

{0x0001 0800}: CMP1 = 0x0800 for two period

provide a burst mode with two periods active every six PWM periods, as shown on Figure 486.

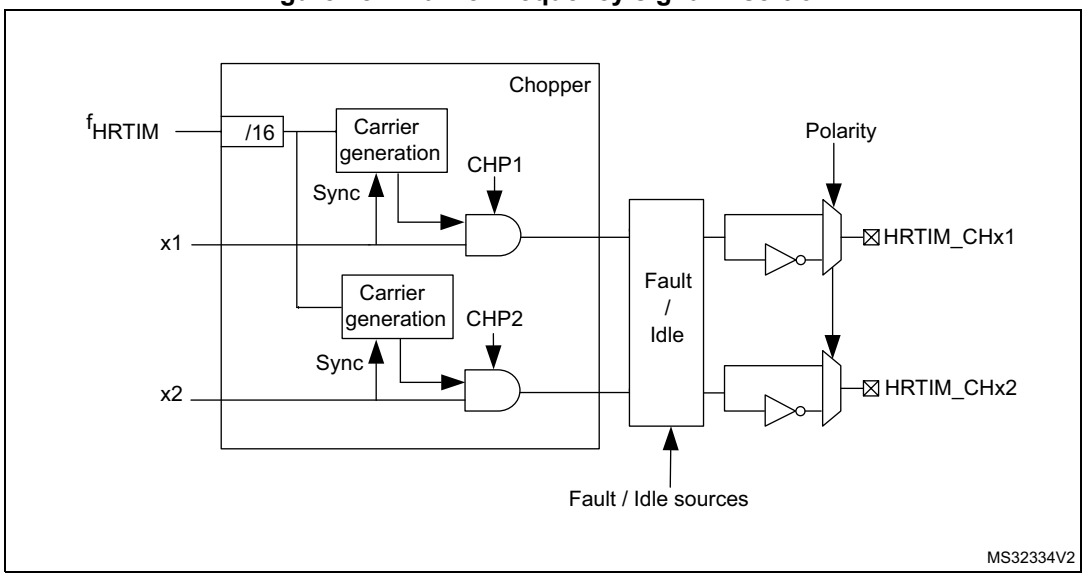
Figure 486. Burst mode emulation example



33.3.16 Chopper

A high-frequency carrier can be added on top of the timing unit output signals to drive isolation transformers. This is done in the output stage before the polarity insertion, as shown on *Figure 487*, using CHP1 and CHP2 bits in the HRTIM_OUTxR register, to enable chopper on outputs 1 and 2, respectively.

Figure 487. Carrier frequency signal insertion



The chopper parameters can be adjusted using the HRIM_CHPxR register, with the possibility to define a specific pulsewidth at the beginning of the pulse, to be followed by a carrier frequency with programmable frequency and duty cycle, as in *Figure 488*.

CARFRQ[3:0] bits define the frequency, ranging from 2.03 MHz to 32.5 MHz (for f_{HRTIM} = 520 MHz) following the formula $F_{CHPFRQ} = f_{HRTIM} / (16 \times (CARFRQ[3:0]+1))$.

The duty cycle can be adjusted by 1/8 step with CARDTY[2:0], from 0/8 up to 7/8 duty cycle. When CARDTY[2:0] = 000 (duty cycle = 0/8), the output waveform only contains the starting pulse following the rising edge of the reference waveform, without any added carrier.

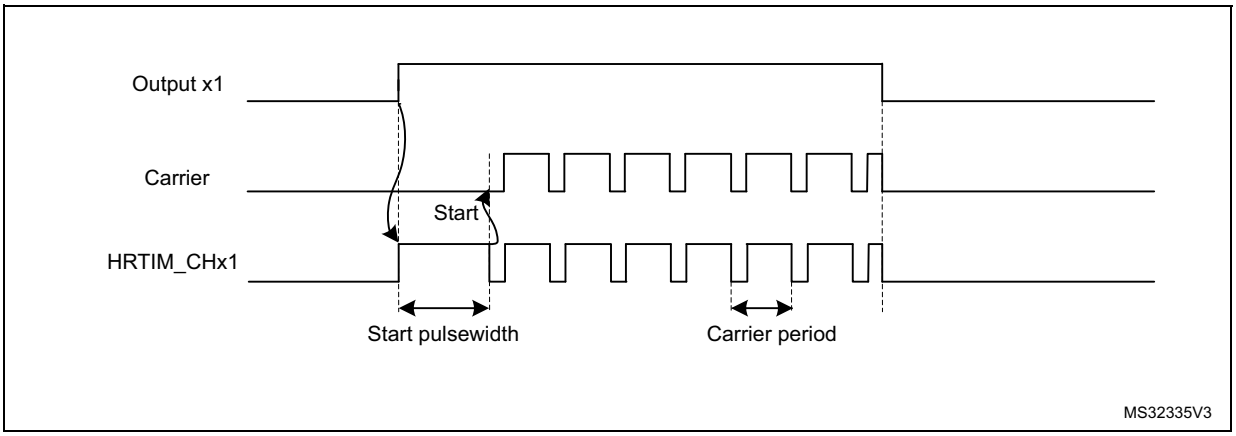
The pulsewidth of the initial pulse is defined using the STRPW[3:0] bitfield as follows: $t_{1STPW} = (STRPW[3:0]+1) \times 16 \times t_{HRTIM}$ and ranges from 31 ns to 492 ns (for f_{HRTIM} = 520 MHz).

The carrier frequency parameters are defined based on the f_{HRTIM} frequency, and do not dependent on the CKPSC[2:0] setting.

In chopper mode, the carrier frequency and the initial pulsewidth are combined with the reference waveform using an AND function. A synchronization is performed at the end of the initial pulse to have a repetitive signal shape.

The chopping signal is stopped at the end of the output waveform active state, without waiting for the current carrier period to be completed. It can thus contain shorter pulses than programmed.

Figure 488. HRTIM outputs with Chopper mode enabled



Note: CHP1 and CHP2 bits must be set prior to the output enable done with TxyOEN bits in the HRTIM_OENR register.

CARFRQ[2:0], CARDTY[2:0], and STRPW[3:0] bitfields cannot be modified while the chopper mode is active (at least one of the two CHPx bits is set).

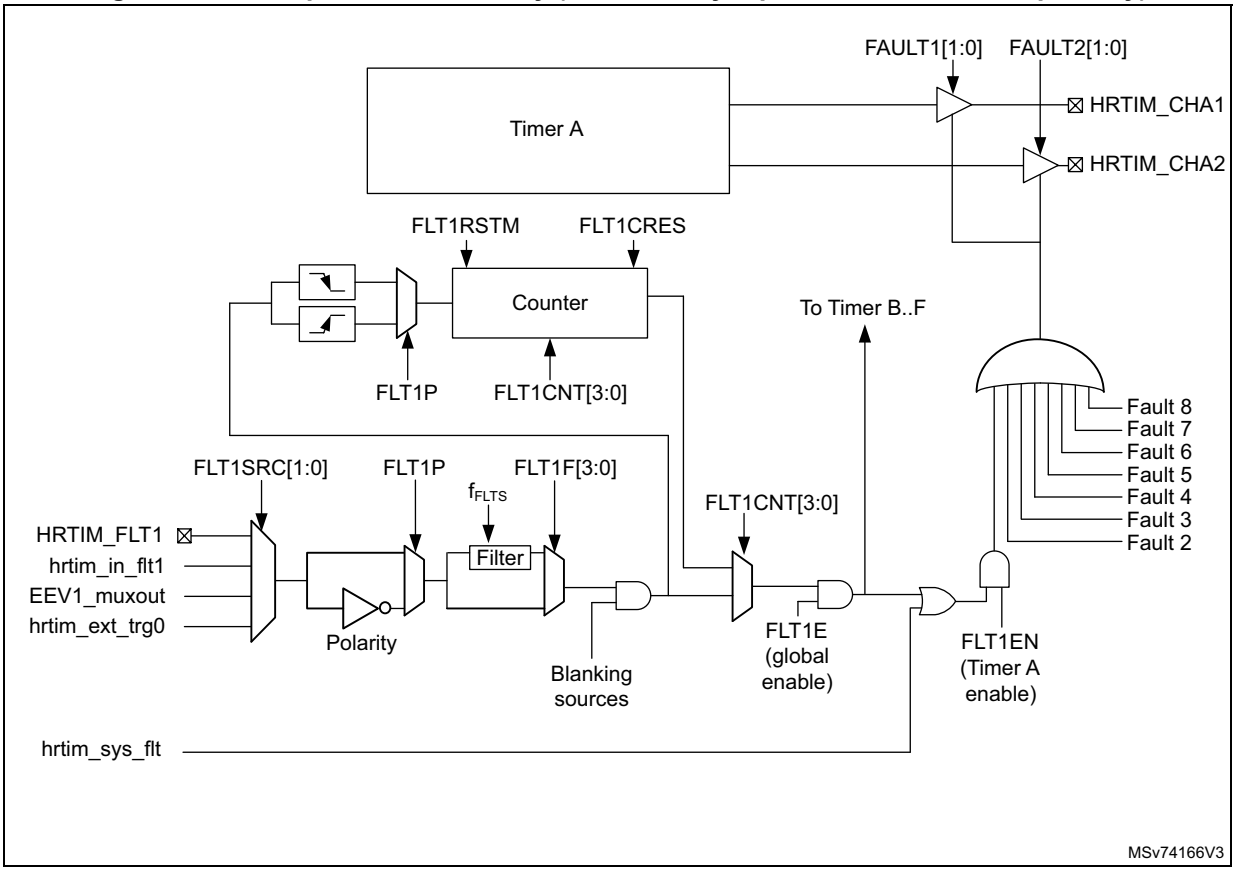
Caution: The chopper mode must only be used with inactive IDLE state (IDLES1 = 0 and IDLES2 = 0).

33.3.17 Fault protection

The HRTIM has a versatile fault protection circuitry to disable the outputs in case of an abnormal operation. Once a fault has been triggered, the outputs take a predefined safe state. This state is maintained until the output is reenabled by software. In case of a permanent fault request, the output remains in its fault state, even if the software attempts to re-enable them, until the fault source disappears.

The HRTIM has eight FAULT input channels; all of them are available and can be combined for each of the eight timing units, as shown on *Figure 489*.

Figure 489. Fault protection circuitry (FAULT1 fully represented, FAULT2..8 partially)



Each fault channel is fully configurable using HRTIM_FLTINR1, HRTIM_FLTINR2 and HRTIM_FLTINR5 registers before being routed to the timing units. FLTxSRC FLTxSRC[1:0] bit selects the source of the fault signal that can be either a digital input or an internal event (built-in comparator output).

Table 333 details the three available sources for each of the six faults channels.

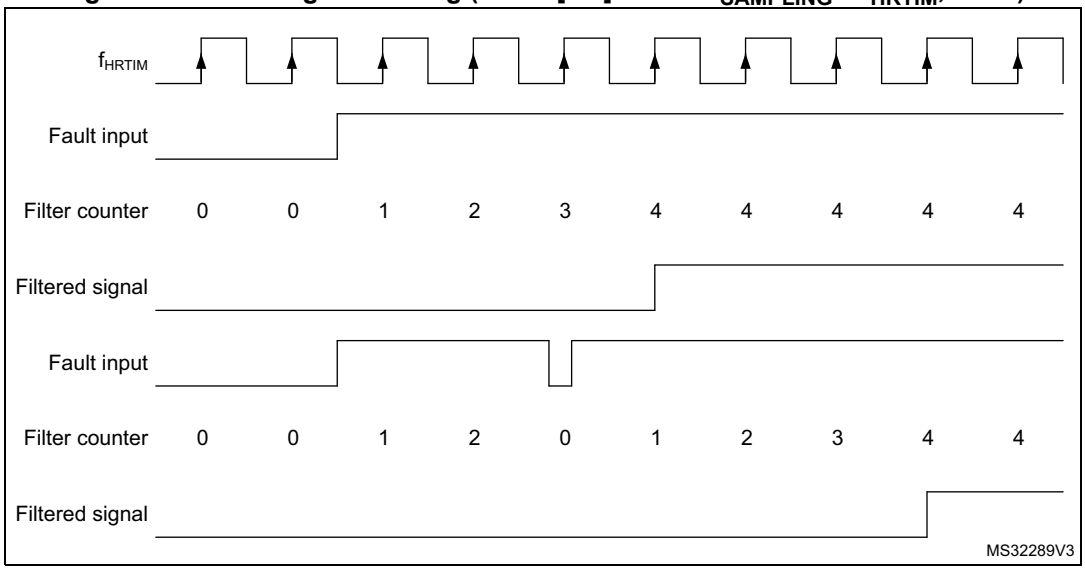
The EEVx_muxout event mentioned in *Figure 489* is taken after the hrtim_eevx[8:1] input multiplexer controlled by the EExSRC[2:0]bits. Refer to *Figure 461* for details.

The polarity of the signal can be selected to define the active level, using the FLTxP polarity bit in HRTIM_FLTINRx registers. If FLTxP = 0, the signal is active at low level; if FLTxP = 1, it is active when high.

The fault information can be filtered after the polarity setting. If FLTxF[3:0] bitfield is set to 0000, the signal is not filtered and acts asynchronously, independently from the f_{HRTIM} clock. For all other FLTxF[3:0] bitfield values, the signal is digitally filtered. The digital filter is made of a counter in which a number N of valid samples are needed to validate a transition on the output. If the input value changes before the counter has reached the value N, the counter is reset and the transition is discarded (considered as a spurious event). If the counter reaches N, the transition is considered as valid and transmitted as a correct external event. Consequently, the digital filter adds a latency to the external events being filtered,

depending on the sampling clock and on the filter length (number of valid samples expected). *Figure 490* shows how a spurious fault signal is filtered.

Figure 490. Fault signal filtering (FLTxF[3:0]= 0010: $f_{SAMPLING} = f_{HRTIM}$, N = 4)



The filtering period ranges from two cycles of the f_{HRTIM} clock up to eight cycles of the f_{FLTS} clock divided by 32. f_{FLTS} is defined using FLTSD[1:0] bits in the HRTIM_FLTINR2 register. *Table 357* summarizes the sampling rate and the filter length. A jitter of 1 sampling clock period must be subtracted from the filter length to take into account the uncertainty due to the sampling and have the effective filtering.

Table 357. Sampling rate and filter length vs FLTFxF[3:0] and clock setting

-	f_{FLTS} vs FLTSD[1:0]				Filter length for f_{HRTIM} = 520 MHz	
FLTFxF[3:0]	**00**	**01**	**10**	**11**	**Min**	**Max**
0001,0010,0011	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}, N =2 3.8 ns	f_{HRTIM}, N =8 15.4 ns
0100, 0101	f_{HRTIM} /2	f_{HRTIM} /4	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /2, N = 6 23 ns	f_{HRTIM} /16, N = 8 246 ns
0110, 0111	f_{HRTIM} /4	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /4, N = 6 46 ns	f_{HRTIM} /32, N = 8 492 ns
1000, 1001	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /8, N = 6 92 ns	f_{HRTIM} /64, N = 8 0.98 µs
1010, 1011, 1100	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /128	f_{HRTIM} /16, N = 5 154 ns	f_{HRTIM} /128, N = 8 1.97 µs
1101, 1110, 1111	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /128	f_{HRTIM} /256	f_{HRTIM} /32, N = 5 308 ns	f_{HRTIM} /256, N = 8 3.94 µs

Fault blanking and event counting

The fault inputs can be temporarily disabled to blank spurious fault events. The blanking sources are listed in the *Table 358* below.

Table 358. Fault input blanking events

-	FLTxBLKS = 0, reset-aligned window		FLTxBLKS = 1 moving window	
Fault input	**Blanking window start**	**Blanking window end**	**Blanking window start**	**Blanking window end**
FAULT1	Timer A reset/roll-over	Timer A CMP3 event	Timer A CMP4 event	Timer A CMP3 event
FAULT2	Timer B reset/roll-over	Timer B CMP3 event	Timer B CMP4 event	Timer B CMP3 event
FAULT3	Timer C reset/roll-over	Timer C CMP3 event	Timer C CMP4 event	Timer C CMP3 event
FAULT4	Timer D reset/roll-over	Timer D CMP3 event	Timer D CMP4 event	Timer D CMP3 event
FAULT5	Timer E reset/roll-over	Timer E CMP3 event	Timer E CMP4 event	Timer E CMP3 event
FAULT6	Timer F reset/roll-over	Timer F CMP3 event	Timer F CMP4 event	Timer F CMP3 event
FAULT7	Timer D reset/roll-over	Timer D CMP1 event	Timer D CMP2 event	Timer D CMP1 event
FAULT8	Timer F reset/roll-over	Timer F CMP1 event	Timer F CMP2 event	Timer F CMP1 event

A fault counter is also used to discard multiple spurious fault events and define an acceptance criteria.

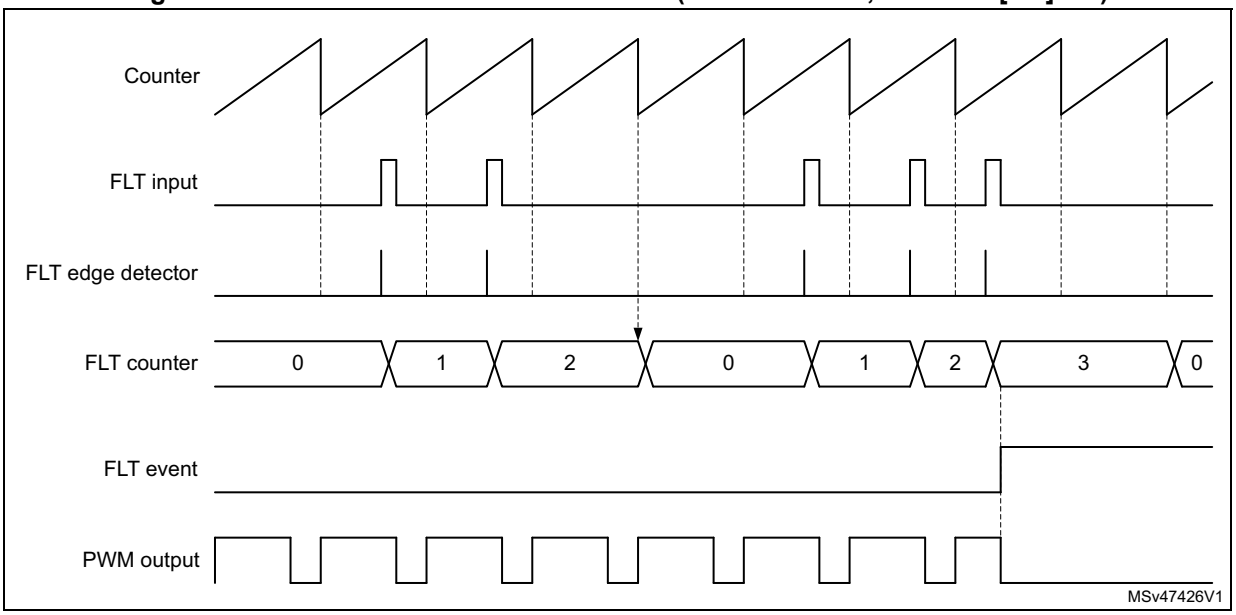
The FLTxCNT[3:0] bitfield selects the FAULTx counter threshold. A fault is considered valid when the number of events is equal to the FLTxCNT[3:0] value.

This FLTxRSTM selects the FAULTx counter reset mode

- 0: fault counter hardware reset on reset/roll-over event, as per the *Table 359* below.
- 1: fault counter reset on each reset/roll-over event only if no event occurs during the last counting period, as shown on *Figure 491* below.

The fault counter can be reset by software with the FLTxCRES bit at any time.

Figure 491. Fault counter cumulative mode (FLTxRSTM = 1, FLTxCNT[3:0] = 2)



A given FLTx input counter can be reset by a single source. The *Table 359* indicates which timer unit is associated with a given fault. This does not prevent having a fault line shared by multiple timers (for example FLT1 with event counter enabled, acting on timer A, timer B, and timer C simultaneously).

Table 359. Fault 1..6 counter reset source

Fault Input	Fault counter reset source
hrtim_flt1[4:1]	Timer A reset/roll-over
hrtim_flt2[4:1]	Timer B reset/roll-over
hrtim_flt3[4:1]	Timer C reset/roll-over
hrtim_flt4[4:1]	Timer D reset/roll-over
hrtim_flt5[4:1]	Timer E reset/roll-over
hrtim_flt6[4:1]	Timer F reset/roll-over
hrtim_flt7[4:1]	Timer D reset/roll-over
hrtim_flt8[4:1]	Timer F reset/roll-over

System fault input (hrtim_sys_flt)

This fault is provided by the MCU Class B circuitry (see *Section 10: System configuration, boot and security (SBS)* for details) and corresponds to a system fault coming from:

- The clock security system
- The AXISRAM1 ECC double error detection
- The AXISRAM2 ECC double error detection
- The AXISRAM3 ECC double error detection
- The AXISRAM4 ECC double error detection
- The ITCM ECC double error detection
- The DTCM ECC double error detection
- The DCACHE ECC double error detection
- The Backup RAM ECC double error detection
- The flash memory ECC double error detection
- The PVD detector
- The Cortex®-M7 lockup signal

This input overrides the FAULT inputs and disables all outputs having FAULTy[1:0] = 01, 10, 11.

For each FAULT channel, a write-once FLTxLCK bit in the HRTIM_FLTxR register is used to lock FLTxE, FLTxP, FLTxSRC, FLTxF[3:0] bits (it renders them read-only), for functional safety purpose. If enabled, the fault conditioning set-up is frozen until the next HRTIM or system reset.

Once the fault signal is conditioned as explained above, it is routed to the timing units. For any of them, the six fault channels are enabled using bits FLT1EN to FLT6EN in the HRTIM_FLTxR register, and they can be selected simultaneously (the sysfault is automatically enabled as long as the output is protected by the fault mechanism). This is used to have, for instance:

- One fault channel simultaneously disabling several timing units
- Multiple fault channels being ORed to disable a single timing unit

A write-once FLTLCK bit in the HRTIM_FLTxR register is used to lock FLTxEN bits (it renders them read-only) until the next reset, for functional safety purpose. If enabled, the timing unit fault-related set-up is frozen until the next HRTIM or system reset.

For each of the timers, the output state during a fault is defined with FAULT1[1:0] and FAULT2[1:0] bits in the HRTIM_OUTxR register (see *Section 33.3.14*).

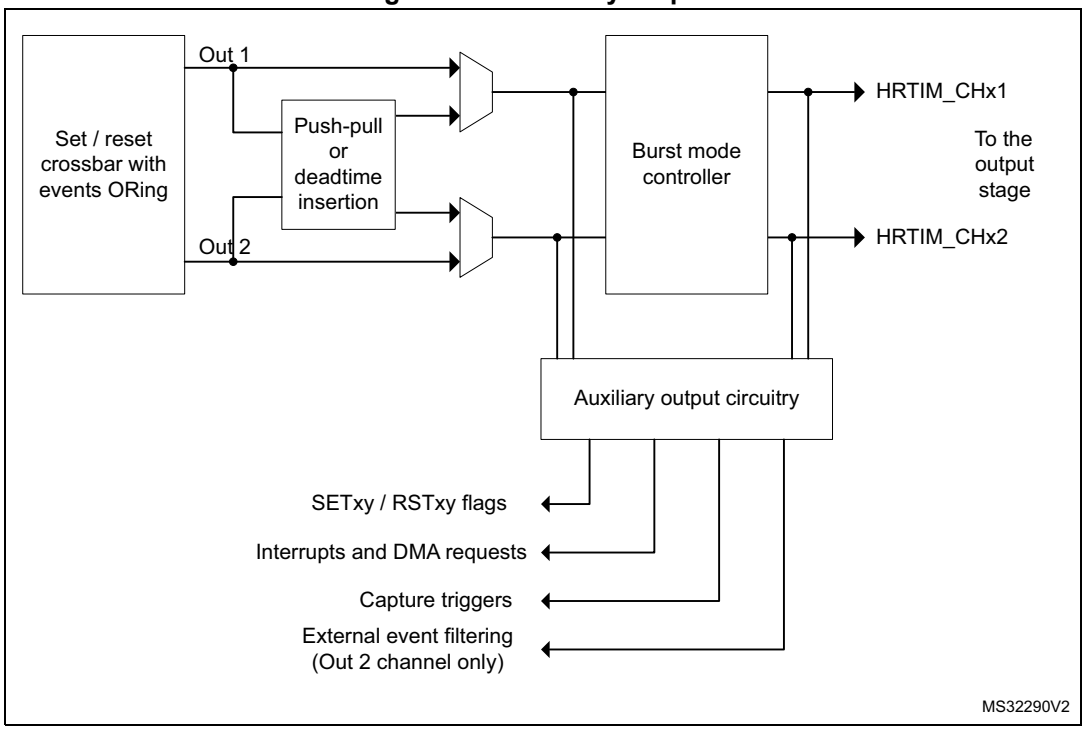
33.3.18 Auxiliary outputs

Timer A to E have auxiliary outputs in parallel with the regular outputs going to the output stage. They provide the following internal status, events, and signals:

- SETxy and RSTxy status flags, together with the corresponding interrupts and DMA requests
- Capture triggers upon output set/reset
- External event filters following a Tx2 output copy (see details in *Section 33.3.9*)

The auxiliary outputs are taken either before or after the burst mode controller, depending on the HRTIM operating mode. An overview is given on *Figure 492*.

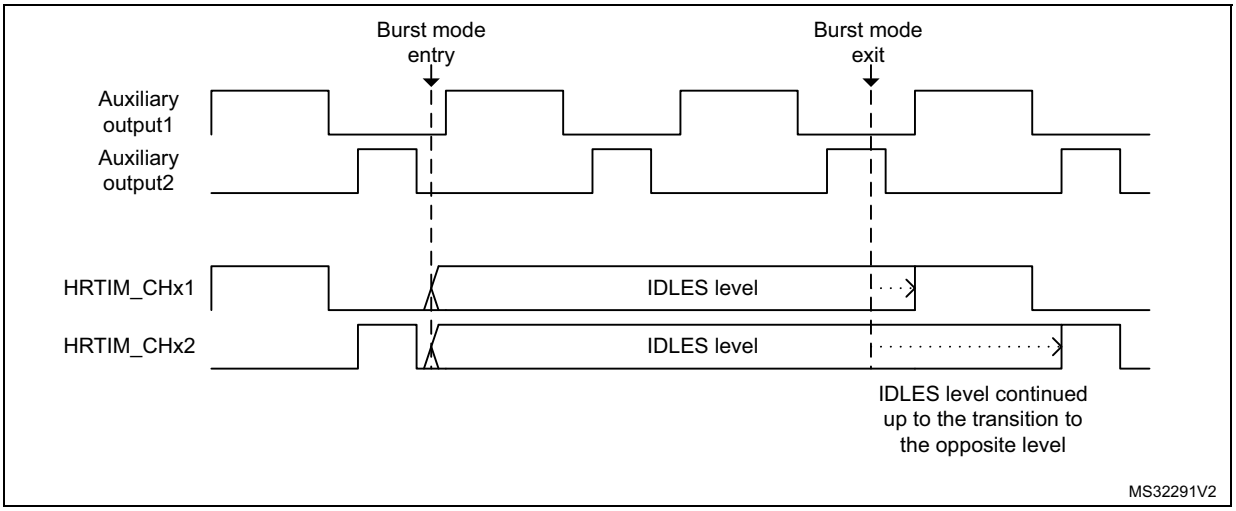
Figure 492. Auxiliary outputs



By default, the auxiliary outputs are copies of outputs Tx1 and Tx2. The exceptions are:

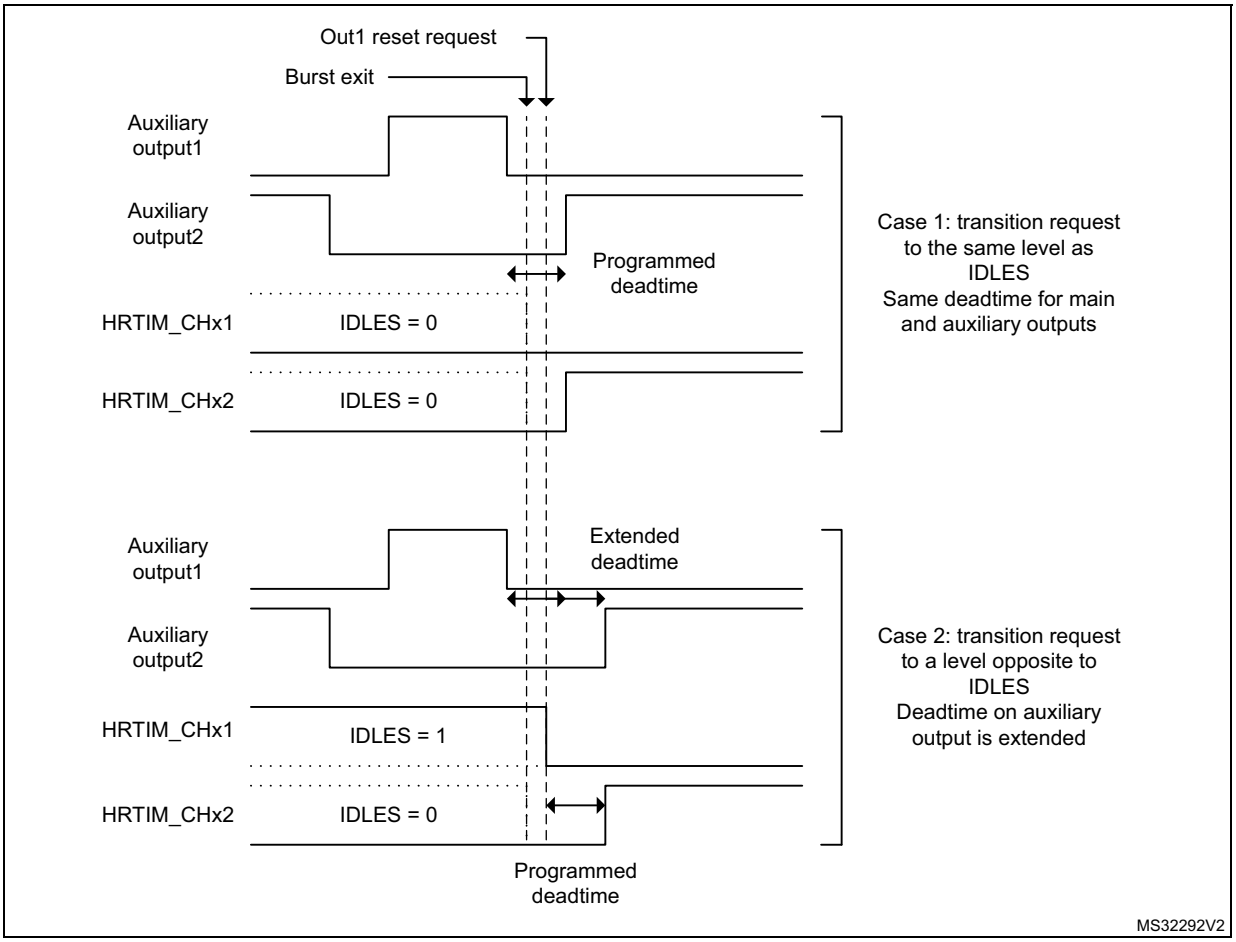
- The delayed idle and the balanced idle protections, when the deadtime is disabled (DTEN = 0). When the protection is triggered, the auxiliary outputs are maintained and follow the signal coming out of the crossbar. On the contrary, if the deadtime is enabled (DTEN = 1), both main and auxiliary outputs are forced to an inactive level.
- The burst mode (TCEN=1, IDLEMx=1); there are two cases:
  a) If DTEN=0 or DIDLx=0, the auxiliary outputs are not affected by the burst mode entry and continue to follow the reference signal coming out of the crossbar (see *Figure 493*).
  b) If the deadtime is enabled (DTEN=1) together with the delayed burst mode entry (DIDLx=1), the auxiliary outputs have the same behavior as the main outputs. They are forced to the IDLES level after a deadtime duration, then they keep this level during all the burst period. When the burst mode is terminated, the IDLES level is maintained until a transition occurs to the opposite level, similarly to the main output.

Figure 493. Auxiliary and main outputs during burst mode (DIDLx = 0)



The signal on the auxiliary output can be slightly distorted when exiting from the burst mode or when reenabling the outputs after a delayed protection, if this happens during a deadtime. In this case, the deadtime applied to the auxiliary outputs is extended so that the deadtime on the main outputs is respected. *Figure 494* gives some examples.

Figure 494. Deadtime distortion on auxiliary output when exiting burst mode



33.3.19 Synchronizing the HRTIM with other timers or HRTIM instances

The HRTIM provides options for synchronizing multiple HRTIM instances, as a master unit (generating a synchronization signal) or as a slave (waiting for a trigger to be synchronized). This feature can also be used to synchronize the HRTIM with other timers, either external or on-chip. The synchronization circuitry is controlled inside the master timer.

Synchronization output

This section explains how the HRTIM must be configured to synchronize external resources and act as a master unit.

Four events can be selected as the source to be sent to the synchronization output. This is done using SYNCSRC[1:0] bits in the HRTIM_MCR register, as follows:

- 00: master timer start
  This event is generated when MCEN bit is set or when the timer is restarted after having reached the period value in single-shot mode. It is also generated on a reset which occurs during the counting (when CONT or RETRIG bits are set).
- 01: master timer compare 1 event
- 10: timer A start
  This event is generated when TACEN bit is set or when the counter is reset and restarts counting in response to this reset. The following counter reset events are not propagated to the synchronization output: counter roll-over in continuous mode, and discarded reset request in single-shot nonretriggerable mode. The reset is only taken into account when it occurs during the counting (CONT or RETRIG bits are set).
- 11: timer A compare 1 event

SYNCOUT[1:0] bits in the HRTIM_MCR register specify how the synchronization event is generated. When SYNCOUT[1:0] bits are set to 01, the synchronization event is generated internally.

The synchronization pulses can be generated on the HRTIM_SCOUT output pin, with SYNCOUT[1:0] = 1x. SYNCOUT[0] bit specifies the polarity of the synchronization signal. If SYNCOUT[0] = 0, the HRTIM_SCOUT pin has a low idle level and issues a positive pulse of 16 f_{HRTIM} clock cycles length for the synchronization). If SYNCOUT[0] = 1, the idle level is high and a negative pulse is generated.

Note: The synchronization pulse is followed by an idle level of 16 f_{HRTIM} clock cycles during which any new synchronization request is discarded. Consequently, the maximum synchronization frequency is $f_{HRTIM}/32$.

The idle level on the HRTIM_SCOUT pin is applied as soon as the SYNCOUT[1:0] bits are enabled (i.e. the bitfield value is different from 00).

The synchronization output initialization procedure must be done prior to the configuration of the MCU outputs and counter enable, in the following order:

1. SYNCOUT[1:0] and SYNCSRC[1:0] bitfield configuration in HRTIM_MCR
2. HRTIM_SCOUT pin configuration (see the “General-purpose I/Os” section)
3. Master or timer A counter enable (MCEN or TACEN bit set)

When the synchronization input mode is enabled and starts the counter (using SYNCSTRTM/SYNCSTRTx bits) simultaneously with the synchronization output mode (SYNCSRC[1:0] = 00 or 10), the output pulse is generated only when the counter is starting

or is reset while running. Any reset request clearing the counter without causing it to start does not affect the synchronization output.

Synchronization input

The HRTIM can be synchronized by external sources, as per the programming of the SYNCIN[1:0] bits in the HRTIM_MCR register:

- 00: synchronization input is disabled
- 01: another on-chip HRTIM instance
- 10: the On-chip timer TRGO output connected to hrtim_in_sync[2] input (refer to *Table 325* for details).
- 11: a positive pulse on the HRTIM_SCIN input pin (hrtim_in_sync[3])

This bitfield cannot be changed once the destination timer (master timer or timing unit) is enabled (MCEN and/or TxCEN bit set).

The HRTIM_SCIN input is rising-edge sensitive. The timer behavior is defined with the following bits present in HRTIM_MCR and HRTIM_TIMxCR registers (see *Table 360* for details):

- Synchronous start: the incoming signal starts the timer's counter (SYNCSTRTM and/or SYNCSTRTx bits set). TxCEN (MCEN) bits must be set to have the timer enabled and the counter ready to start. In continuous mode, the counter does not start until the synchronization signal is received.
- Synchronous reset: the incoming signal resets the counter (SYNCRSTM and/or SYNCRSTx bits set). This event decrements the repetition counter as any other reset event.

The synchronization events are taken into account only once the related counters are enabled (MCEN or TxCEN bit set). A synchronization request triggers a SYNC interrupt.

Note: A synchronized start event resets the counter if the current counter value is above the active period value.

The effect of the synchronization event depends on the timer operating mode, as summarized in *Table 360*.

Table 360. Effect of sync event versus timer operating modes

Operating mode	SYNC RSTx	SYNC STRTx	Behavior following a SYNC reset or start event
Single-shot non-retriggerable	0	1	Start events are taken into account when the counter is stopped and: – once the MCEN or TxCEN bits are set – once the period has been reached. A start occurring when the counter is stopped at the period value resets the counter. A reset request clears the counter but does not start it (the counter can solely be re-started with the synchronization). Any reset occurring during the counting is ignored (as during regular non-retriggerable mode).
	1	X	Reset events are starting the timer counting. They are taken into account only if the counter is stopped and: – once the MCEN or TxCEN bits are set – once the period has been reached. When multiple reset requests are selected (from HRTIM_SCIN and from internal events), only the first arriving request is taken into account.

Table 360. Effect of sync event versus timer operating modes (continued)

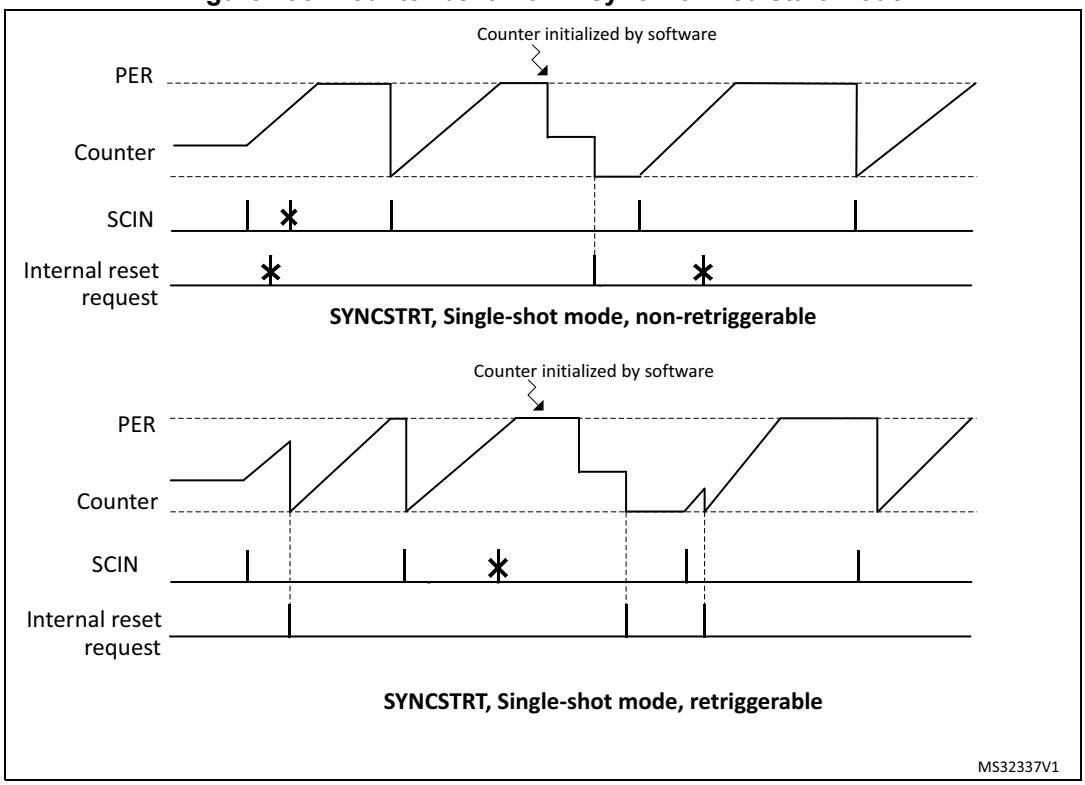
Operating mode	SYNC RSTx	SYNC STRTx	Behavior following a SYNC reset or start event
Single-shot retriggerable	0	1	The counter start is effective only if the counter is not started or period is elapsed. Any synchronization event occurring after counter start has no effect. A start occurring when the counter is stopped at the period value resets the counter. A reset request clears the counter but does not start it (the counter can solely be started by the synchronization). A reset occurring during counting is taken into account (as during regular retriggerable mode).
	1	X	The reset from HRTIM_SCIN is taken into account as any HRTIM's timer counter reset from internal events and is starting or re-starting the timer counting. When multiple reset requests are selected, the first arriving request is taken into account.
Continuous mode	0	1	The timer is enabled (MCEN or TxCEN bit set) and is waiting for the synchronization event to start the counter. Any synchronization event occurring after the counter start has no effect (the counter can solely be started by the synchronization). A reset request clears the counter but does not start it.
	1	X	The reset from HRTIM_SCIN is taken into account as any HRTIM's timer counter reset from internal events and is starting or re-starting the timer counting. When multiple reset requests are selected, the first arriving request is taken into account.

Note: When multiple HRTIMs are synchronously started, the HRTIM which initiates the synchronization is started one f_{HRTIM} *clock cycle after its SYNC output signal is sent. This is used to compensate for the internal propagation delay and to have the subsequent HRTIM timer cycle accurately synchronized.*

When multiple HRTIMs are synchronously reset, the slave timer is reset one f_{HRTIM} *clock cycle after the master request (internal propagation delay).*

Figure 495 presents how the synchronized start is done in single-shot mode.

Figure 495. Counter behavior in synchronized start mode



Remote update disable

The remote update disable bits RMUDIS and RTxUDIS in HRTIM_CR1 register can temporarily disable the update event on other on-chip HRTIMER.

Both the regular and remote update disable bits are located in HRTIM_CR1 register, so that both HRTIMERs can be simultaneously disabled.

This mechanism is used to make sure that both HRTIMER registers are updated consistently, without any risk to have one HRTIMER updated before the other.

Any pending update is ignored while the update disable bits are set. The update happens on the next valid update event, once the update disable bits have been cleared.

For security and task isolation reasons, to prevent one HRTIMER to disable updates on a another HRTIMER, the remote update disable authorization bits RMUDISA and RTxUDISA in HRTIM_CR3 must first be set. By default, the remote update disable functionality is inactive.

33.3.20 ADC triggers

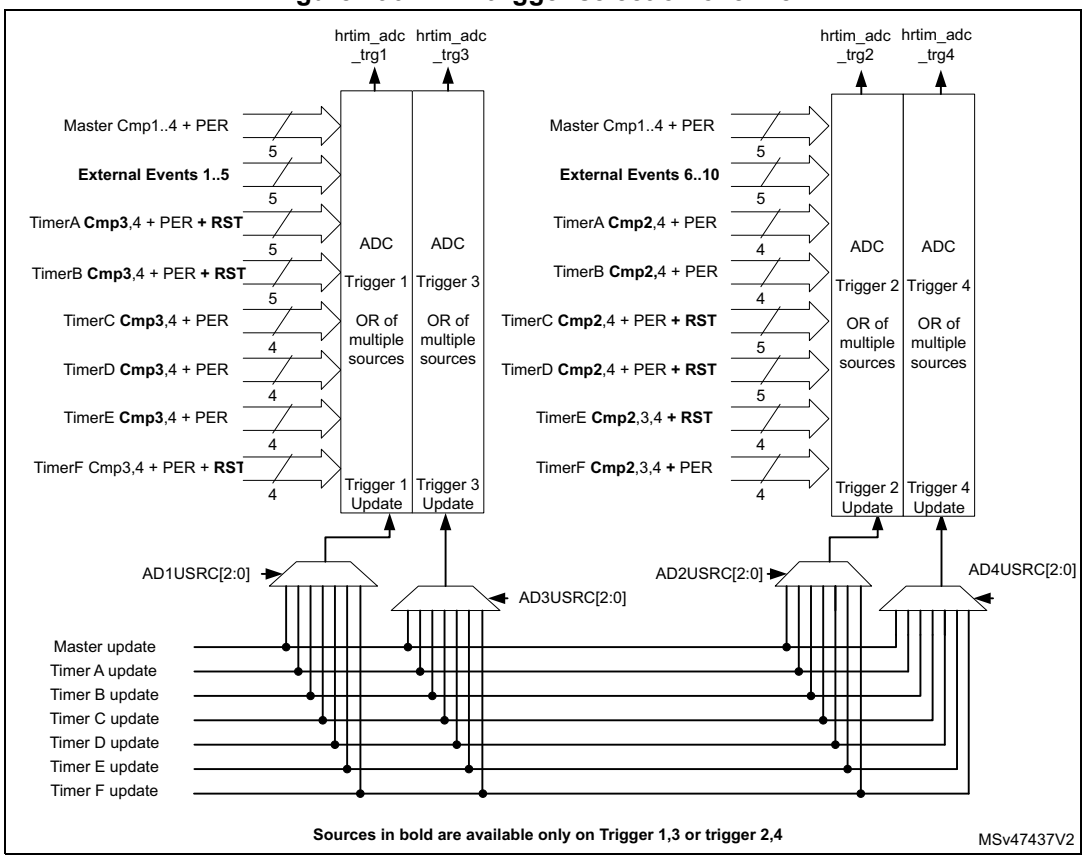
The ADCs can be triggered by the master and the six timing units.

10 independent triggers are available for both the regular and the injected sequencers of the ADCs. The external events can be used as triggers. They are taken right after the

conditioning defined in the HRTIM_EECRx registers, and are not depending on the EEFxR1 and EEFxR2 register settings.

Up to 32 events can be combined (ORed) for ADC triggers 1 to 4, in HRTIM_ADC1R to HRTIM_ADC4R registers, as shown on *Figure 496*. The ADC triggers 1/3 and 2/4 are using the same source set. A multiple triggering is possible within a single switching period by selecting several sources simultaneously. A typical use case is for a nonoverlapping multiphase converter, where all phases can be sampled in a row using a single ADC trigger output.

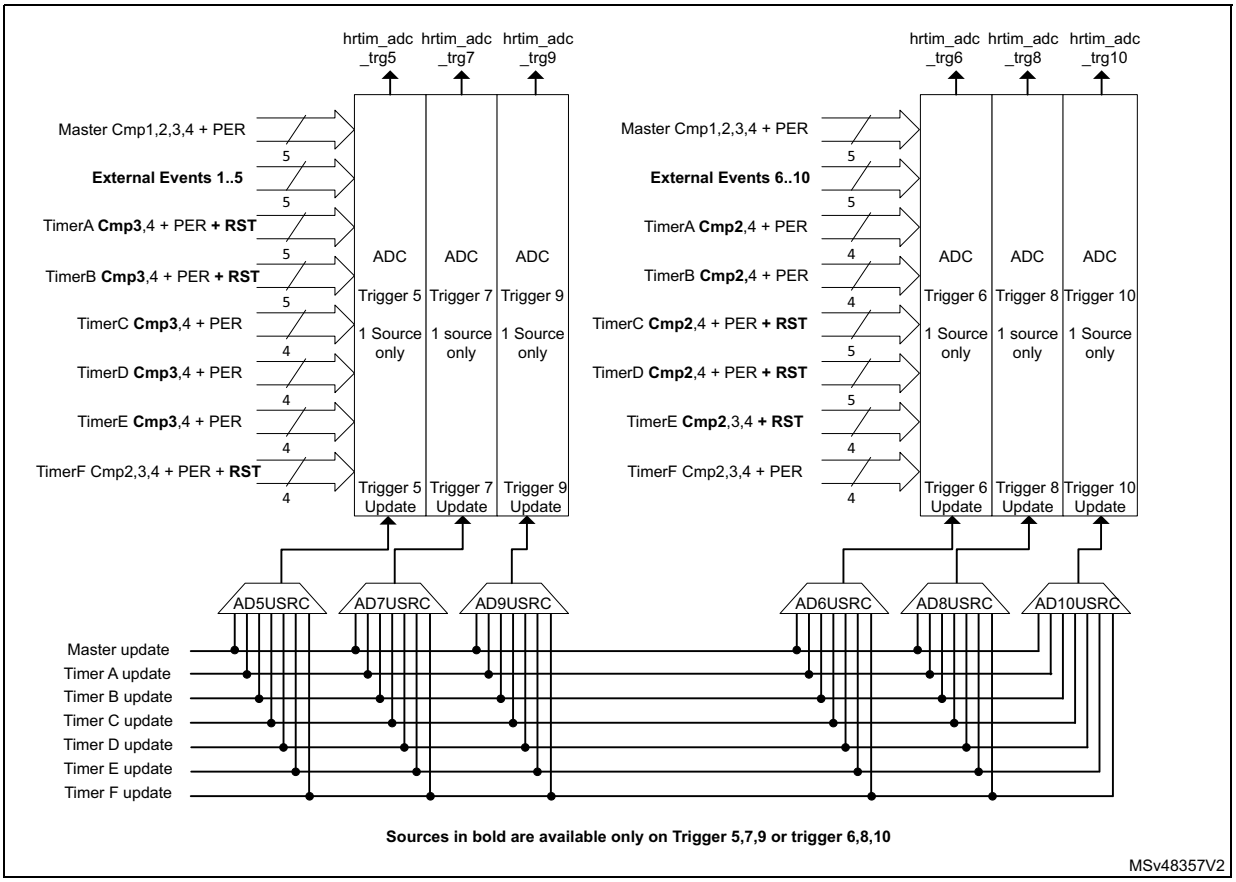
Figure 496. ADC trigger selection overview



The ADC triggers 5 to 10 are configured in the HRTIM_ADCER register, as shown on *Figure 497*. The ADC triggers 5/7/9 and 6/8/10 are using the same source set.

A single source can be selected at once for these triggers (1 out of 32 possible events).

Figure 497. ADC triggers



HRTIM_ADC1R to HRTIM_ADC4R and HRTIM_ADCER registers are preloaded and can be updated synchronously with the timer they are related to. The update sources are defined with ADxUSRC[2:0] bits in the HRTIM_CR1 and HRTIM_ADCUR registers.

For instance, if ADC trigger 1 outputs timer A CMP2 events (HRTIM_ADC1R = 0x0000 0400), HRTIM_ADC1R is typically updated simultaneously with timer A (AD1USRC[2:0] = 001).

When the preload is disabled (PREEN bit reset) in the source timer, the HRTIM_ADCxR registers are not preloaded either: a write access results in an immediate update of the trigger source.

ADC postscaler

A postscaling unit is used to reduce the ADC trigger rate with respect to the switching frequency.

Each ADC trigger rate can be individually adjusted using the ADCxPSC[4:0] bits in the HRTIM_ADCxPS1 and HRTIM_ADCxPS2 registers as shown in *Figure 498* below for the up-counting mode.

In the center-aligned mode, the ADC postscaling is done on events selected with the ADROM[1:0] bitfield, programmed in the source timer, as shown in *Figure 499*. The

ADROM[1:0] bitfield is coding for any event that can trigger the ADC: reset, roll-over (period) and compare event:

- ADROM[1:0] = 00: event generated both during up and down-counting phases
- ADROM[1:0] = 01: event generated during down-counting phases
- ADROM[1:0] = 10: event generated during up-counting phases

The ADC postscaler programming registers are preloaded and can be updated on-the-fly without stopping the timers.

Figure 498. ADC trigger post-scaling in up-counting mode

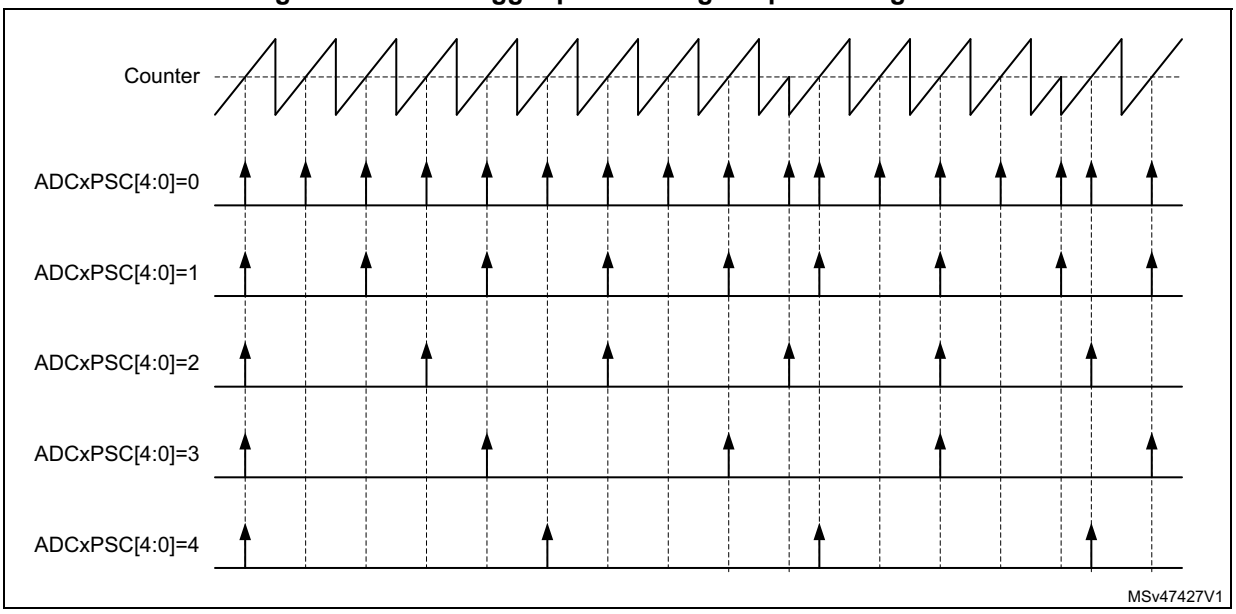
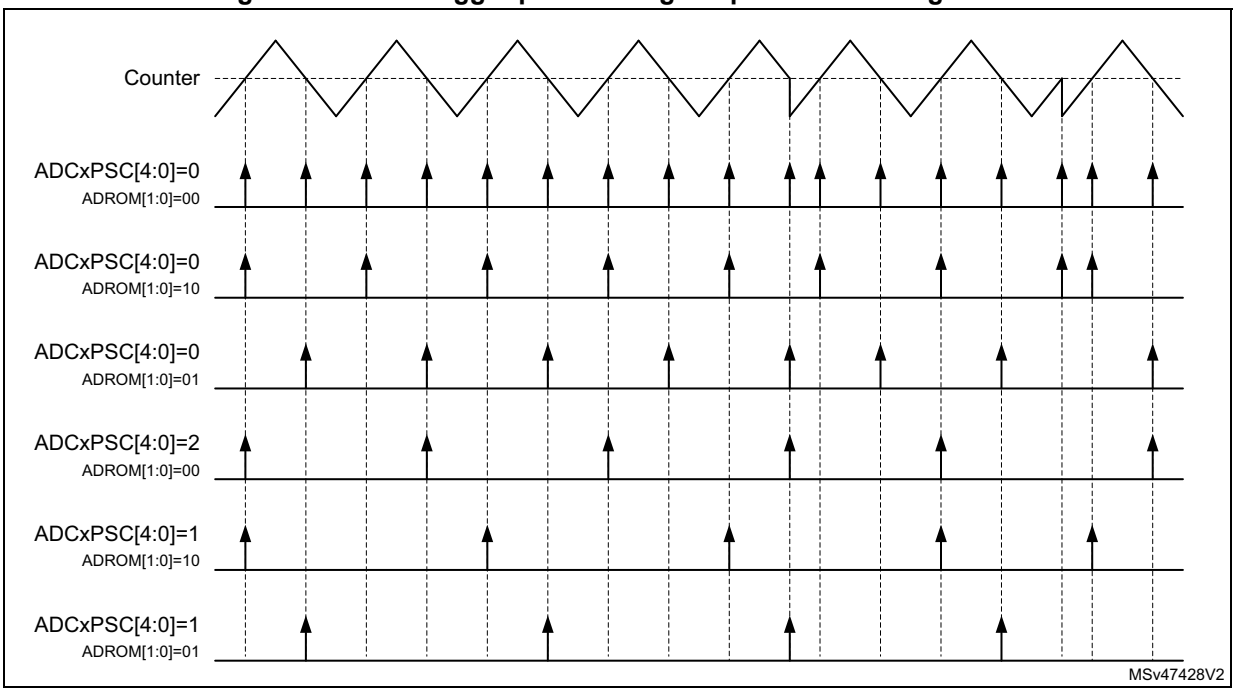


Figure 499. ADC trigger post-scaling in up/down counting mode



33.3.21 DAC triggers

The HRTIM is used to have the embedded DACs updated synchronously with the timer updates.

The update events from the master timer and the timer units can generate DAC update triggers on any of the three hrtim_dac_trgx outputs.

Note: Each timer has its own DAC-related control register.

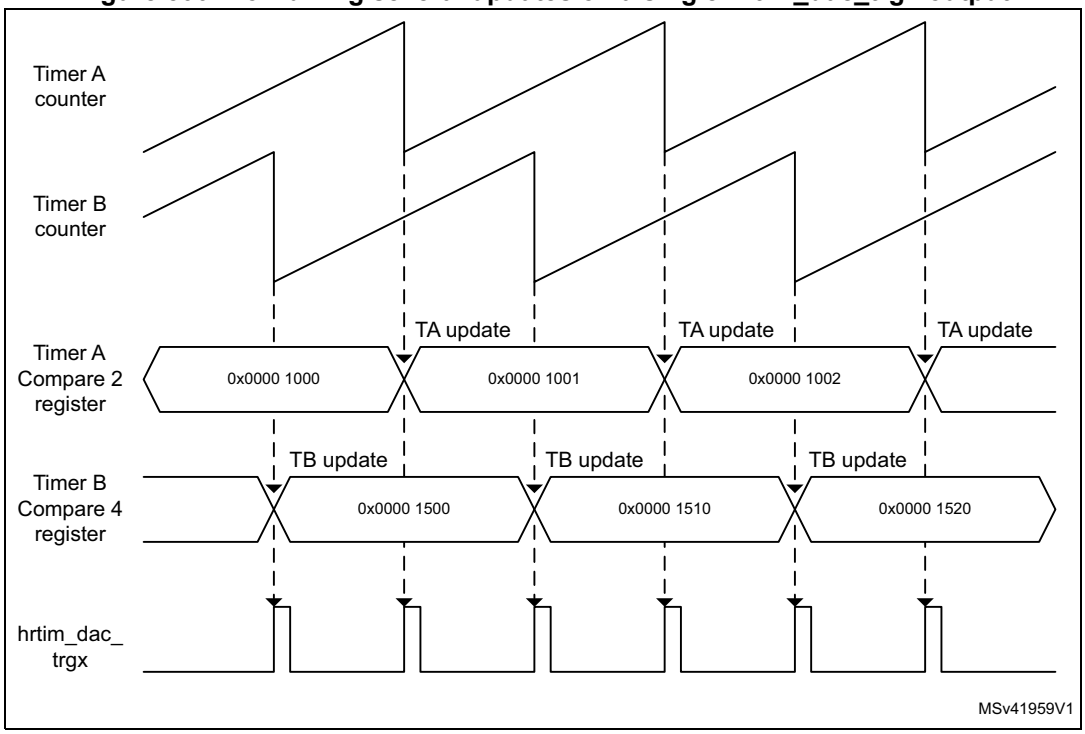
DACSYNC[1:0] bits of the HRTIM_MCR and HRTIM_TIMxCR registers are programmed as follows:

- 00: No update generated
- 01: Update generated on hrtim_dac_trg1
- 10: Update generated on hrtim_dac_trg2
- 11: Update generated on hrtim_dac_trg3

An output pulse of one f_{HRTIM} clock period is generated on the hrtim_dac_trgx output.

When DACSYNC[1:0] bits are enabled in multiple timers, the hrtim_dac_trgx output consists of an OR of all timers' update events. For instance, if DACSYNC = 1 in timer A and in timer B, the update event in timer A is ORed with the update event in timer B to generate a DAC update trigger on the corresponding hrtim_dac_trgx output, as shown on *Figure 500*.

Figure 500. Combining several updates on a single hrtim_dac_trgx output



Refer to *Table 335: HRTIM1/HRTIM2 DAC triggers connections* for connections to the DACs.

Dual channel DAC trigger

Slope compensation techniques and hysteretic control to be easily implemented using HRTIM built-in features and the DAC sawtooth generator. The principle is to have a DAC generating a decreasing saw-tooth synchronized with the PWM period, or a square wave synchronized with PWM signal.

This mode is enabled with the DCDE bit in the TIMxCR2 register. This bit cannot be changed once the timer is operating (TxEN bit set).

It uses two trigger outputs, as shown on the *Figure 501* below:

- The hrtim_dac_reset_trgx generates DAC reset/update events
- The hrtim_dac_step_trgx generates requests for incremental DAC value changes

The DCDR bit in the TIMxCR2 register defines when the hrtim_dac_reset_trgx trigger is generated:

- DCDR = 0: the trigger is generated on counter reset or roll-over event
- DCDR = 1: the trigger is generated on output 1 set event

Note: The DCDR bit is not significant when the DCDE bit is reset (Dual channel DAC trigger disabled).

The DCDS bit in the TIMxCR2 register defines when the hrtim_dac_step_trgx trigger is generated:

- DCDS = 0: the trigger is generated on compare 2 event
- DCDS = 1: the trigger is generated on output 1 reset event

The DCDR and DCDS bits allows the following use cases to be covered:

- Edge-aligned slope compensation (DCDR = DCDS = 0): the DAC's sawtooth starts on PWM period beginning and multiple triggers are generated during the period
- Center-aligned slope compensation (DCDR = 1 DCDS = 0): the DAC's sawtooth starts on the output set event and multiple triggers are generated during the period
- Hysteretic controller: the DAC value must be changed twice per period, when the output state changes. 2 triggers are generated per PWM period. In edge-aligned mode (DCDR=0, DCDS =1), the triggers are generated on counter reset or roll-over. In center-aligned mode (DCDR=1, DCDS=1), the triggers are generated when the output is set.

The compare 2 has a particular operating mode when the DCDE is set and the DCDS bit is reset. The active comparison value is automatically updated as soon as a compare match has occurred, so that the trigger can be repeated periodically with a period equal to the CMP2 value, as represented on *Figure 501*.
The dual channel DAC trigger with DCDS bit reset (compare 2 event used) must not be used simultaneously with modes using CMP2 (triple/quad interleaved and triggered-half modes).

Note: The CMP2 value can be changed on-the-fly. The new value is taken into account on the next coming compare match.

Note: When the DCDS bit is reset, the CMP2 value must not be modified by other mechanisms: the interleaved, triggered half and balanced idle modes must be disabled.

Table 361 below gives an example, for generating six triggers within a PWM period. It shows that it is necessary to round up the division result to the upper value.

Let's consider a counter period TIMxPER = 8192. Dividing 8192 by 6 yields 1365.33.

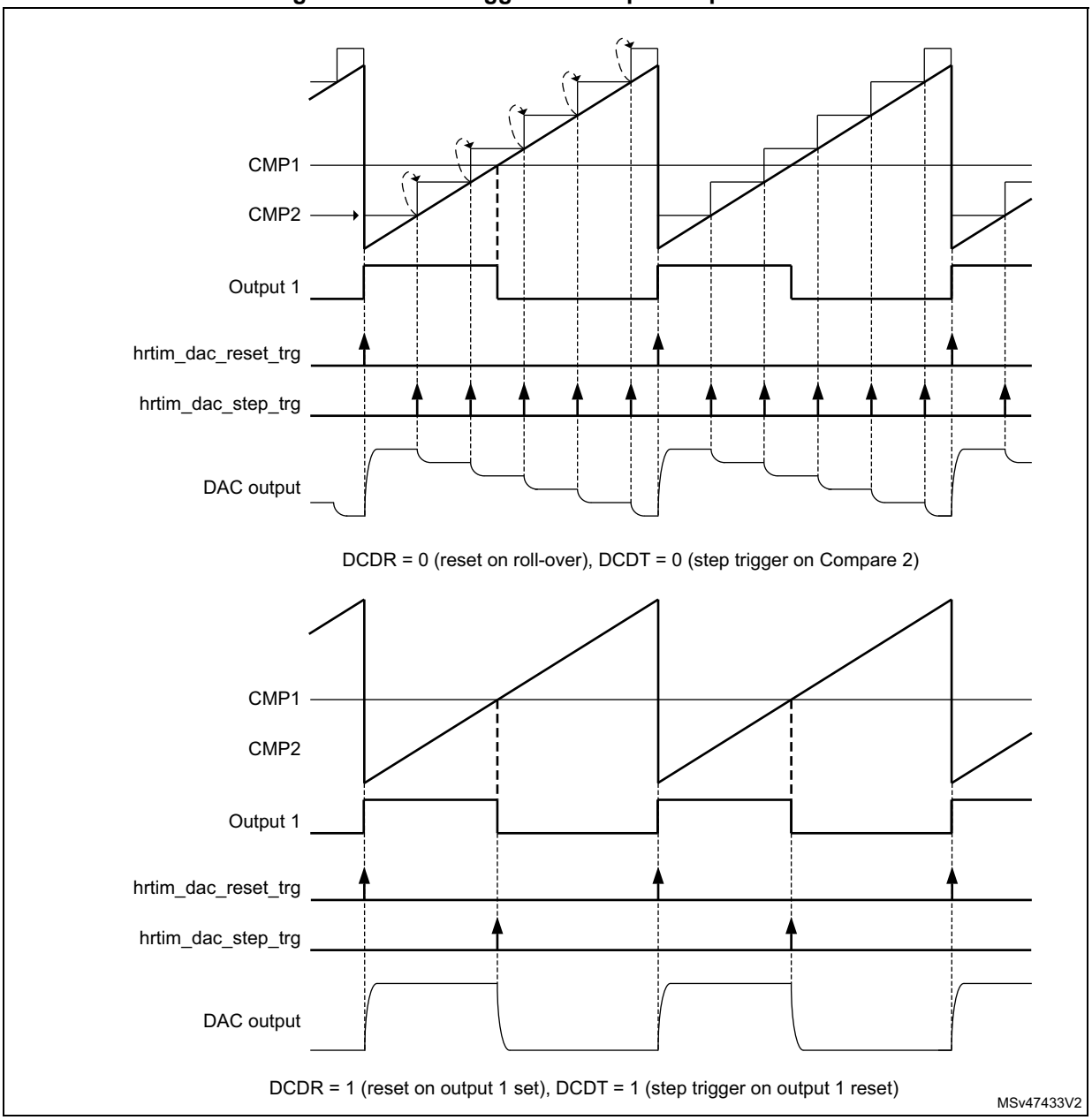
- Round down value: 1365: 7 triggers are generated, the 6th and 7th being very close (respectively for counter = 8190 and 8192)
- Round up value:1366: six triggers are generated. The sixth trigger on hrtim_dac_step_trg (for counter = 8192) is aborted by the counter roll-over from 8192 to 0.

Table 361. DAC dual channel trigger example

-	**CMP2 = 1365**	**hrtim_dac_ reset_trg**	**hrtim_dac_ step_trg**	**CMP2 = 1366**	**hrtim_dac_ reset_trg**	**hrtim_dac_ step_trg**
Counter value	1365	-	1	1366	-	1
	2730	-	2	2732	-	2
	4095	-	3	4098	-	3
	5460	-	4	5464	-	4
	6825	-	5	6830	-	5
	8190	-	6	8192	6	-
	8192	7	-	1366	-	1
	1365	-	1	2732	-	2
	...	-	-	...	-	-

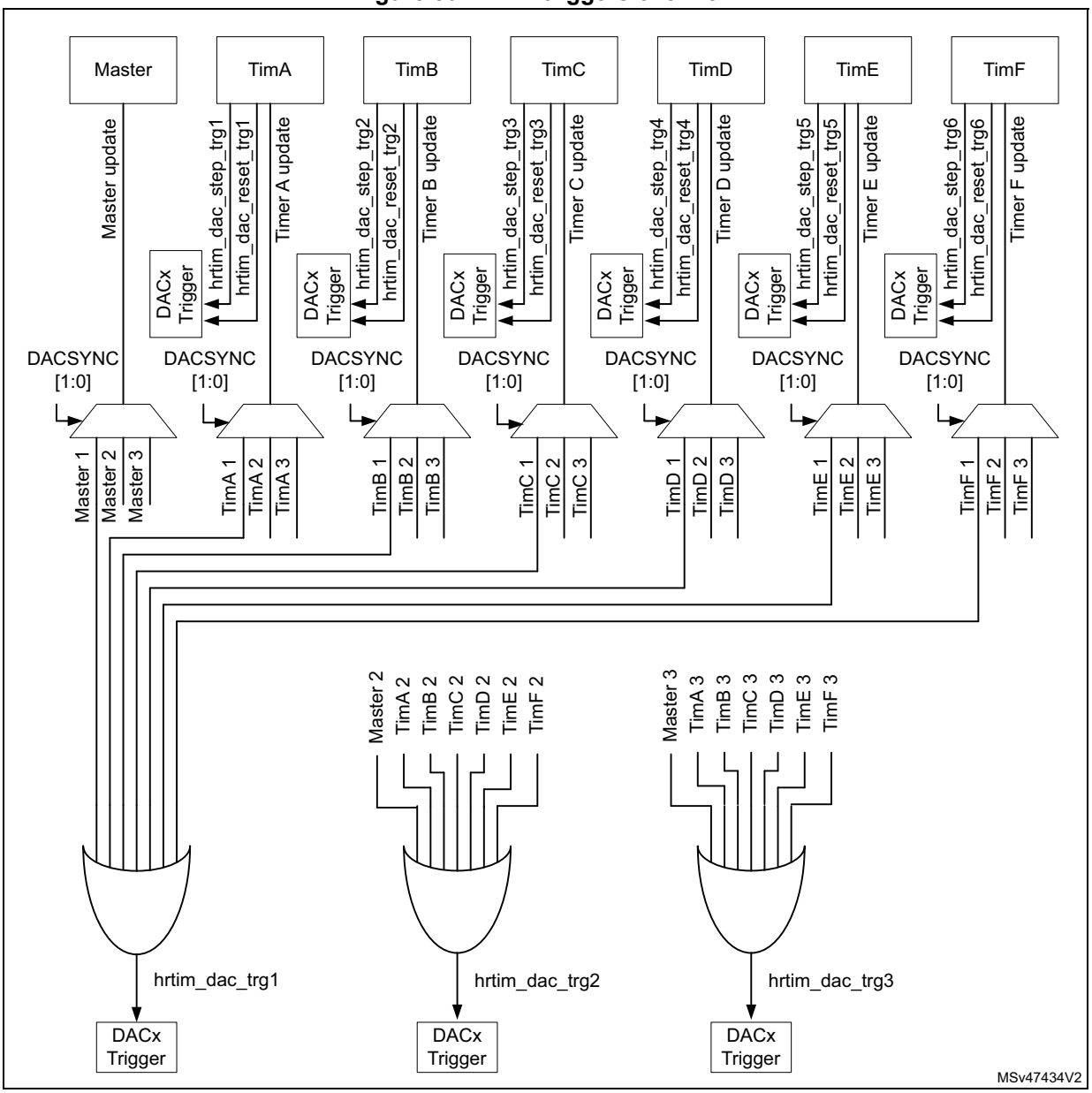
Note: In centered-pattern mode, it is mandatory to have an even number of triggers per switching period, so as to avoid unevenly spaced triggers around counter's peak value.

Figure 501. DAC triggers for slope compensation



The *Figure 502* below provides an overview of all the available DAC triggers.

Figure 502. DAC triggers overview



33.3.22 Interrupts

7 interrupts can be generated by the master timer:

- Master timer registers update
- Synchronization event received
- Master timer repetition event
- Master compare 1 to 4 event

14 interrupts can be generated by each timing unit:

- Crossbar override rearm delay elapsed
- Delayed protection triggered
- Counter reset or roll-over event
- Output 1 and output 2 reset (transition active to inactive)
- Output 1 and output 2 set (transition inactive to active)
- Capture 1 and 2 events
- Timing unit registers update
- Crossbar override trigger
- Repetition event
- Compare 1 to 4 event

8 global interrupts are generated for the whole HRTIM:

- System fault and fault 1 to 6 (regardless of the timing unit attribution)
- Burst mode period completed

The interrupt requests are grouped in eight vectors as follows:

- hrtim_it1: master timer interrupts (master update, sync input, repetition, MCMP1..4) and global interrupt except faults (burst mode period)
- hrtim_it2: TIMA interrupts
- hrtim_it3: TIMB interrupts
- hrtim_it4: TIMC interrupts
- hrtim_it5: TIMD interrupts
- hrtim_it6: TIME interrupts
- hrtim_it7: TIMF interrupts
- hrtim_it8: Dedicated vector all fault interrupts to allow high-priority interrupt handling

Table 362 is a summary of the interrupt requests, their mapping and associated control, and status bits.

Table 362. HRTIM interrupt summary

Interrupt vector	Interrupt event	Event flag	Enable control bit	Flag clearing bit
hrtim_it1	Burst mode period completed	BMPER	BMPERIE	BMPERC
	Master timer registers update	MUPD	MUPDIE	MUPDC
	Synchronization event received	SYNC	SYNCIE	SYNCC
	Master timer repetition event	MREP	MREPIE	MREPC
	Master compare 1 to 4 event	MCMP1	MCMP1IE	MCP1C
		MCMP2	MCMP2IE	MCP2C
		MCMP3	MCMP3IE	MCP3C
		MCMP4	MCMP4IE	MCP4C

Table 362. HRTIM interrupt summary (continued)

Interrupt vector	Interrupt event	Event flag	Enable control bit	Flag clearing bit
hrtim_it2 hrtim_it3 hrtim_it4 hrtim_it5 hrtim_it6 hrtim_it7	Crossbar override rearm delay elapsed	CORDE	CORDEIE	CORDEC
	Delayed protection triggered	DLYPRT	DLYPRTIE	DLYPRTC
	Counter reset or roll-over event	RST	RSTIE	RSTC
	Output 1 and output 2 reset (transition active to inactive)	RSTx1	RSTx1IE	RSTx1C
		RSTx2	RSTx2IE	RSTx2C
	Output 1 and output 2 set (transition inactive to active)	SETx1	SETx1IE	SETx1C
		SETx2	SETx2IE	SETx2C
	Capture 1 and 2 events	CPT1	CPT1IE	CPT1C
		CPT2	CPT2IE	CPT2C
	Timing unit registers update	UPD	UPDIE	UPDC
	Crossbar override trigger	COT	COTIE	COTC
	Repetition event	REP	REPIE	REPC
	Compare 1 to 4 event	CMP1	CMP1IE	CMP1C
		CMP2	CMP2IE	CMP2C
		CMP3	CMP3IE	CMP3C
		CMP4	CMP4IE	CMP4C
hrtim_it8	System fault	SYSFLT	SYSFLTIE	SYSFLTC
	Fault 1 to 6	FLT1	FLT1IE	FLT1C
		FLT2	FLT2IE	FLT2C
		FLT3	FLT3IE	FLT3C
		FLT4	FLT4IE	FLT4C
		FLT5	FLT5IE	FLT5C
		FLT6	FLT6IE	FLT6C

33.3.23 DMA

Most of the events able to generate an interrupt can also generate a DMA request, even both simultaneously. Each timer (master, TIMA...F) has its own DMA enable register.

The individual DMA requests are ORed into seven channels as follows:

- 1 channel for the master timer
- 1 channel per timing unit (TIMA...F)

Note: Before disabling a DMA channel (DMA enable bit reset in TIMxDIER), it is necessary to disable first the DMA controller.

Table 363 is a summary of the events with their associated DMA enable bits.

Table 363. HRTIM DMA request summary

DMA Channel	Event	DMA capable	DMA enable bit
hrtim_dma1 (master timer)	Burst mode period completed	No	N/A
	Master timer registers update	Yes	MUPDDE
	Synchronization event received	Yes	SYNCDE
	Master timer repetition event	Yes	MREPDE
	Master compare 1 to 4 event	Yes	MCMP1DE
		Yes	MCMP2DE
		Yes	MCMP3DE
		Yes	MCMP4DE
hrtim_dma2 (timer A) hrtim_dma3 (timer B) hrtim_dma4 (timer C) hrtim_dma5 (timer D) hrtim_dma6 (timer E) hrtim_dma7 (timer F)	Crossbar override rearm delay elapsed	Yes	CORDEDE
	Delayed protection triggered	Yes	DLYPRTDE
	Counter reset or roll-over event	Yes	RSTDE
	Output 1 and output 2 reset (transition active to inactive)	Yes	RSTx1DE
		Yes	RSTx2DE
	Output 1 and output 2 set (transition inactive to active)	Yes	SETx1DE
		Yes	SETx2DE
	Capture 1 and 2 events	Yes	CPT1DE
		Yes	CPT2DE
	Timing unit registers update	Yes	UPDDE
	Crossbar override trigger	Yes	COTDE
	Repetition event	Yes	REPDE
	Compare 1 to 4 event	Yes	CMP1DE
		Yes	CMP2DE
		Yes	CMP3DE
		Yes	CMP4DE
N/A	System fault	No	N/A
	Fault 1 to 6	No	N/A
	Burst mode period completed	No	N/A

Burst DMA transfers

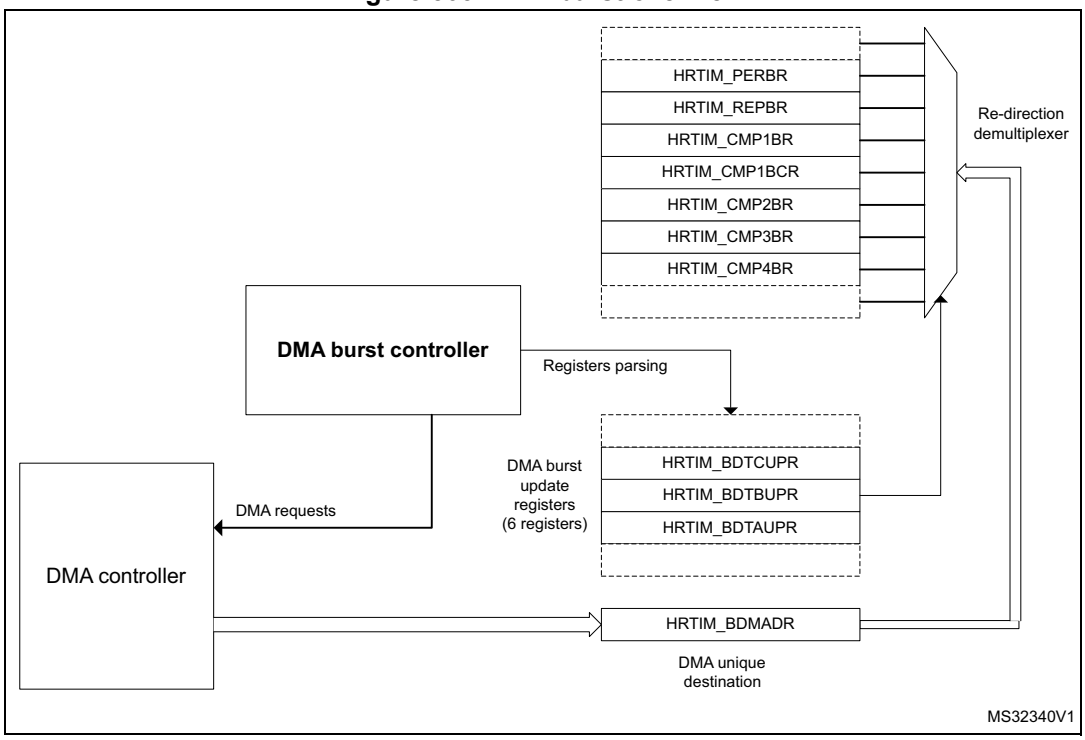
In addition to the standard DMA requests, the HRTIM features a DMA burst controller to have multiple registers updated with a single DMA request. This allows to:

- Update multiple data registers with one DMA channel only,
- Reprogram dynamically one or several timing units, for converters using multiple timer outputs.

The burst DMA feature is only available for one DMA channel, but any of the six channels can be selected for burst DMA transfers.

The principle is to program which registers are to be written by DMA. The master timer and TIMA..E have the burst DMA update register, where most of their control and data registers are associated with a selection bit: HRTIM_BDMUPR, HRTIM_BDTAUPR to HRTIM_BDTEUPR (this is applicable only for registers with write accesses). A redirection mechanism is used to forward the DMA write accesses to the HRTIM registers automatically, as shown on *Figure 503*.

Figure 503. DMA burst overview



When the DMA trigger occurs, the HRTIM generates multiple 32-bit DMA requests and parses the update register. If the control bit is set, the write access is redirected to the associated register. If the bit is reset, the register update is skipped and the register parsing is resumed until a new bit set is detected, to trigger a new request. Once the six update registers (HRTIM_BDMUPR, 5x HRTIM_BDTxUPR) are parsed, the burst is completed and the system is ready for another DMA trigger (see the flowchart on *Figure 504*).

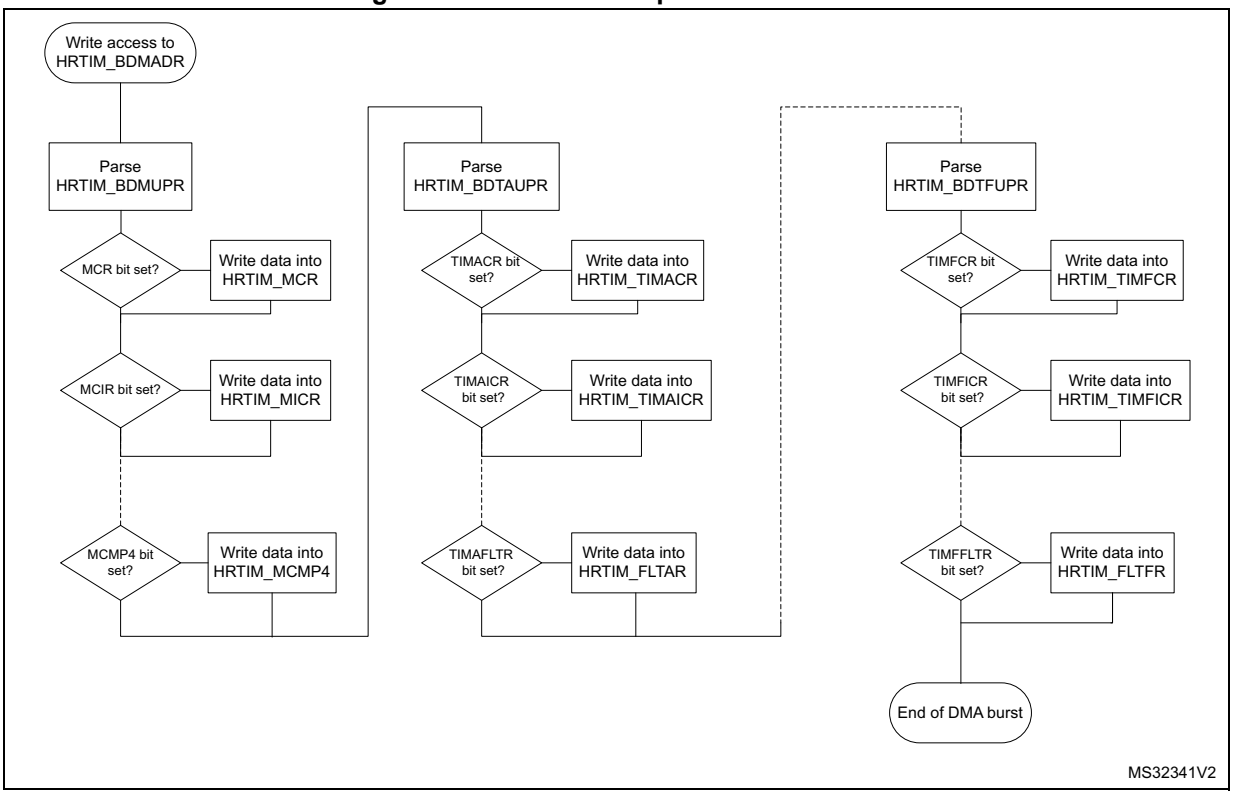
Note: Any trigger occurring while the burst is ongoing is discarded, except if it occurs during the very last data transfer.

The burst DMA mode is permanently enabled (there is no enabled bit). A burst DMA operation is started by the first write access into the HRTIM_BDMADR register.

It is only necessary to have the DMA controller pointing to the HRTIM_BDMADR register as the destination, in the memory, to the peripheral configuration with the peripheral increment mode disabled (the HRTIM handles internally the data rerouting to the final destination register).

To reinitialize the burst DMA mode if it was interrupted during a transaction, it is necessary to write at least to one of the six update registers.

Figure 504. Burst DMA operation flowchart



Several options are available once the DMA burst is completed, depending on the register update strategy.

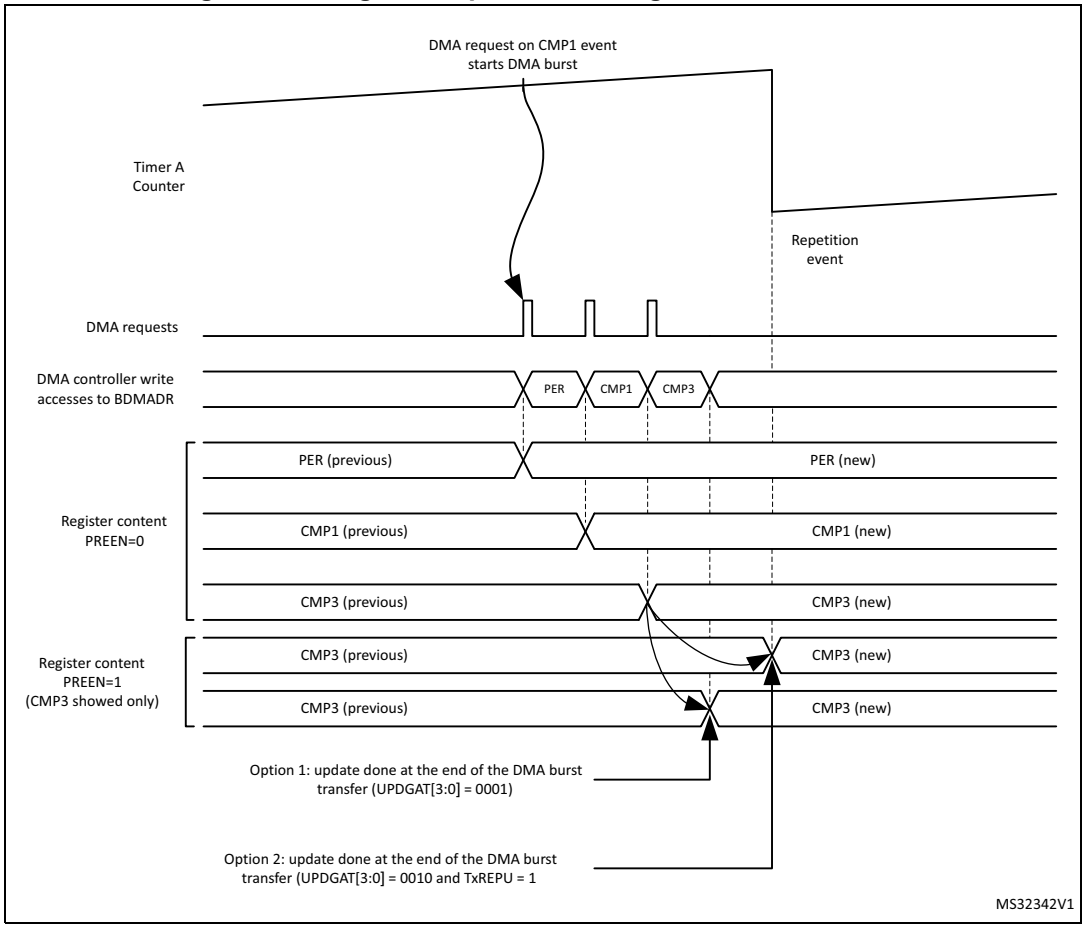
If the PREEN bit is reset (preload disabled), the value written by the DMA is immediately transferred into the active register and the registers are updated sequentially, following the DMA transaction pace.

When the preload is enabled (PREEN bit set), there are three use cases:

1. The update is done independently from DMA burst transfers (UPDGAT[3:0] = 0000 in HRTIM_TIMxCR and BRSTDMA[1:0] = 00 in HRTIM_MCR). In this case, and if it is necessary to have all transferred data taken into account simultaneously, the user must check that the DMA burst is completed before the update event takes place. On the contrary, if the update event happens while the DMA transfer is ongoing, only part of the registers is loaded and the complete register update requires two consecutive update events.
2. The update is done when the DMA burst transfer is completed (UPDGAT[3:0] = 0000 in HRTIM_TIMxCR and BRSTDMA[1:0] = 01 in HRTIM_MCR). This mode guarantees that all new register values are transferred simultaneously. This is done independently from the counter value and can be combined with regular update events, if necessary (for instance, an update on a counter reset when TxRSTU is set).
3. The update is done on the update event following the DMA burst transfer completion (UPDGAT[3:0] = 0010 in HRTIM_TIMxCR and BRSTDMA[1:0] = 10 in HRTIM_MCR). This mode guarantees both a coherent update of all transferred data and the synchronization with regular update events, with the timer counter. In this case, if a regular update request occurs while the transfer is ongoing, it is discarded and the effective update happens on the next coming update request.

The chronogram on *Figure 505* presents the active register content for three cases: PREEN=0, UPDGAT[3:0] = 0001 and UPDGAT[3:0] = 0001 (when PREEN = 1).

Figure 505. Registers update following DMA burst transfer



33.3.24 HRTIM initialization

This section describes the recommended HRTIM initialization procedure, including other related MCU peripherals.

The HRTIM clock source must be enabled in the reset and clock control unit (RCC).

The HRTIM control registers can be initialized as per the power converter topology and the timing units use case. All inputs have to be configured (source, polarity, edge-sensitivity).

The HRTIM outputs must be set up eventually, with the following sequence:

- The polarity must be defined using POLx bits in HRTIM_OUTxR
- The FAULT and IDLE states must be configured using FAULTx[1:0] and IDLESx bits in HRTIM_OUTxR

The HRTIM outputs are ready to be connected to the MCU I/Os. In the GPIO controller, the selected HRTIM I/Os have to be configured as per the alternate function mapping table in the product datasheet.

From this point on, the HRTIM controls the outputs, which are in the IDLE state.

The outputs are configured in RUN mode by setting TxyOEN bits in the HRTIM_OENR register. The two outputs are in the inactive state until the first valid set/reset event in RUN mode. Any output set/reset event (except software requests using SST, SRT) are ignored as long as TxCEN bit is reset, as well as burst mode requests (IDLEM bit value is ignored). Similarly, any counter reset request coming from the burst mode controller is ignored (if TxBM bit is set).

Note: When the deadtime insertion is enabled (DTEN bit set), it is necessary to force the output state by software, using SST and RST bits, to have the outputs in a complementary state as soon as the RUN mode is entered.

The HRTIM operation can eventually be started by setting TxCEN or MCEN bits in HRTIM_MCR.

If the HRTIM peripheral is reset with the reset and clock controller, the HRTIM outputs are put in IDLE mode with a low level. It is recommended to first disconnect the HRTIMER from the outputs (using the GPIO controller) before performing a peripheral reset.

33.3.25 Debug

When a microcontroller enters the debug mode (Cortex®-M7 core halted), the TIMx counter either continues to work normally or stops, depending on DBG_HRTIM_STOP configuration bit in DBG module:

- DBG_HRTIM_STOP = 0: no behavior change, the HRTIM continues to operate.
- DBG_HRTIM_STOP = 1: all HRTIM timers, including the master, are stopped. The outputs in RUN mode enter the FAULT state if FAULTx[1:0] = 01,10,11, or keep their current state if FAULTx[1:0] = 00. The outputs in idle state are maintained in this state. This is permanently maintained even if the MCU exits the halt mode. This is used to maintain a safe state during the execution stepping. The outputs can be enabled again by setting TxyOEN bit (requires the use of the debugger).

Timer behavior during MCU halt when DBG_HRTIM_STOP = 1

The set/reset crossbar, the dead-time and push-pull unit, the idle/balanced fault detection and all the logic driving the normal output in RUN mode are not affected by debug. The output keeps on toggling internally, so as to retrieve regular signals of the outputs when TxyOEN is set again (during or after the MCU halt). Associated triggers and filters are also following internal waveforms when the outputs are disabled.

FAULT inputs and events (any source) are enabled during the MCU halt.

Fault status bits can be set and TxyOEN bits reset during the MCU halt if a fault occurs at that time (TxyOEN and TxyODS are not affected by DBG_HRTIM_STOP bit state).

Synchronization, counter reset, start, and reset-start events are discarded in debug mode, as well as capture events. This is to keep all related registers stable as long as the MCU is halted.

The counter stops counting when a breakpoint is reached. However, the counter enable signal is not reset; consequently no start event is emitted when exiting from debug. All counter reset and capture triggers are disabled, as well as external events (ignored as long as the MCU is halted). The outputs SET and RST flags are frozen, except in case of forced software set/reset. A level-sensitive event is masked during the debug but is active again as soon as the debug is exited. For edge-sensitive events, if the signal is maintained active during the MCU halt, a new edge is not generated when exiting from debug.

The update events are discarded. This prevents any update trigger on hrtim_upd_en[3:1] inputs. DMA triggers are disabled. The burst mode circuit is frozen: the triggers are ignored and the burst mode counter stopped.

33.4 Application use cases

33.4.1 Buck converter

Buck converters are of common use as step-down converters. The HRTIM can control up to 12 buck converters with seven independent switching frequencies.

The converter usually operates at a fixed frequency and the Vin/Vout ratio depends on the duty cycle D applied to the power switch:

$$V_{out} = D \times V_{in}$$

The topology is given on *Figure 506* with the connection to the ADC for voltage reading.

Figure 506. Buck converter topology

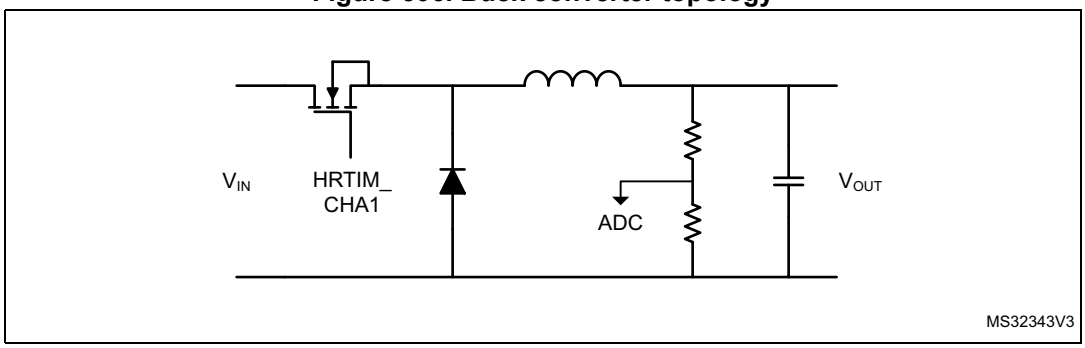
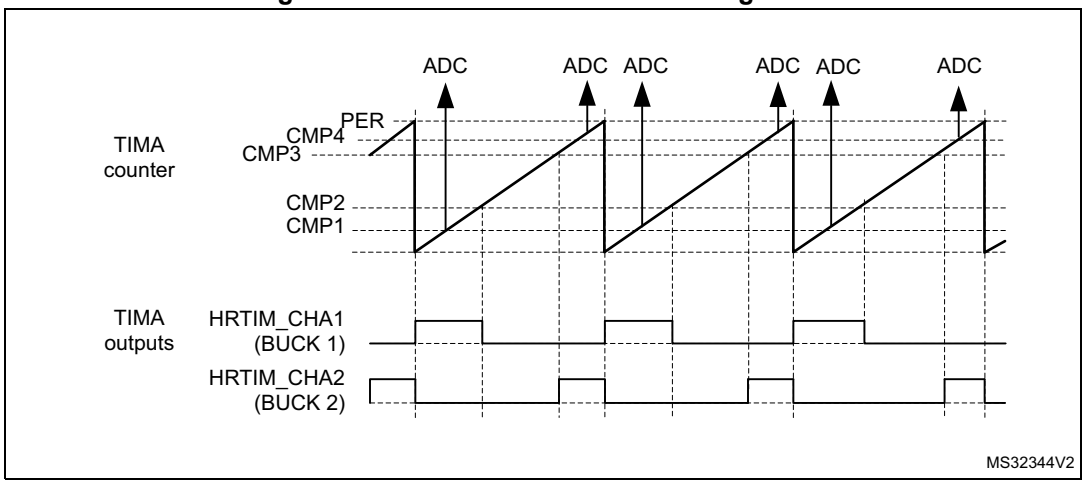


Figure 507 presents the management of two converters with identical frequency PWM signals. The outputs are defined as follows:

- HRTIM_CHA1 set on period, reset on CMP1
- HRTIM_CHA2 set on CMP3, reset on PER

The ADC is triggered twice per period, precisely in the middle of the ON time, using CMP2 and CMP4 events.

Figure 507. Dual Buck converter management

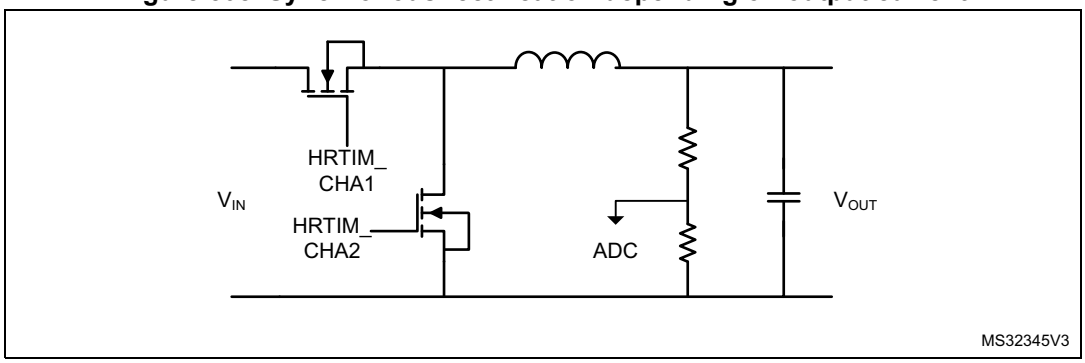


Timers A..E provide either 12 buck converters coupled by pairs (both with identical switching frequencies) or seven completely independent converters (each of them having a different switching frequency), using the master timer as the seventh time base.

33.4.2 Buck converter with synchronous rectification

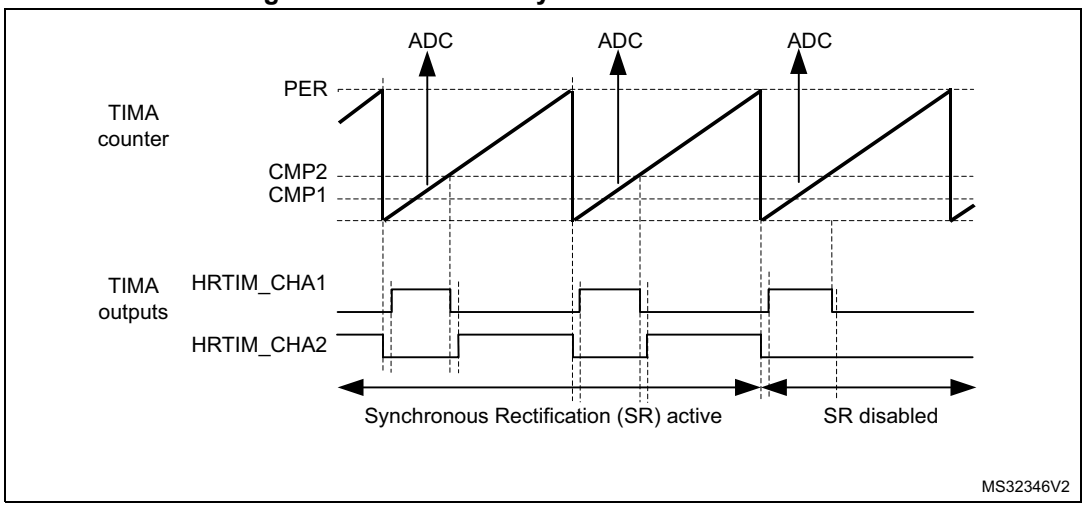
Synchronous rectification is used to minimize losses in buck converters, by means of a FET replacing the freewheeling diode. Synchronous rectification can be turned on or off on the fly depending on the output current level, as shown on *Figure 508*.

Figure 508. Synchronous rectification depending on output current



The main difference versus a single-switch buck converter is the addition of a deadtime for an almost complementary waveform generation on HRTIM_CHA2, based on the reference waveform on HRTIM_CHA1 (see *Figure 509*).

Figure 509. Buck with synchronous rectification



33.4.3 Multiphase converters

Multiphase techniques can be applied to multiple power conversion topologies (buck, flyback). Their main benefits are:

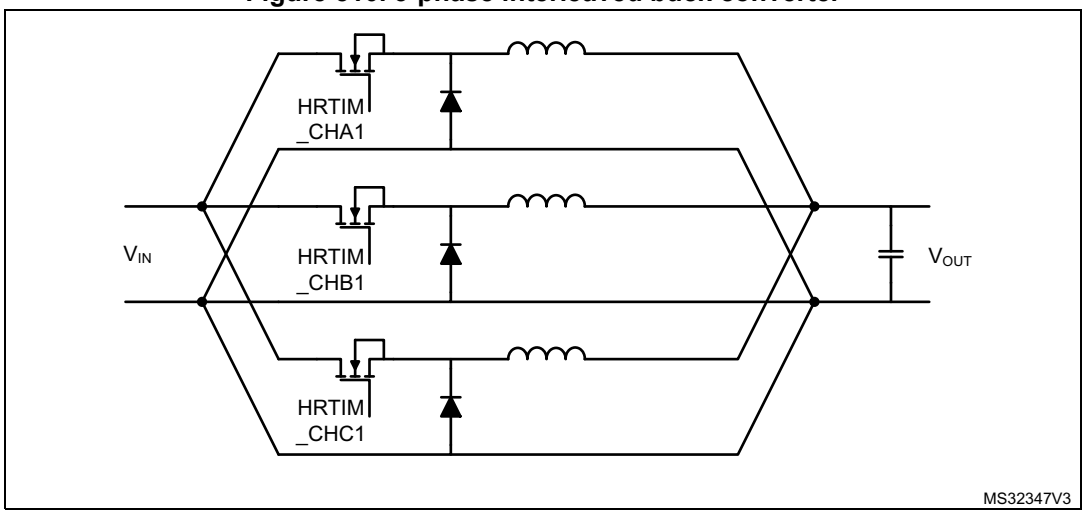
- Reduction of the current ripple on the input and output capacitors
- Reduced EMI
- Higher efficiency at light load by dynamically changing the number of phases (phase shedding)

The HRTIM manages multiple converters. The number of converters that can be controlled depends on the topologies and resources used (including the ADC triggers):

- Five buck converters with synchronous rectification (SR), using the master timer and the five timers
- Four buck converters (without SR), using the master timer and two timers

Figure 510 presents the topology of a 3-phase interleaved buck converter.

Figure 510. 3-phase interleaved buck converter



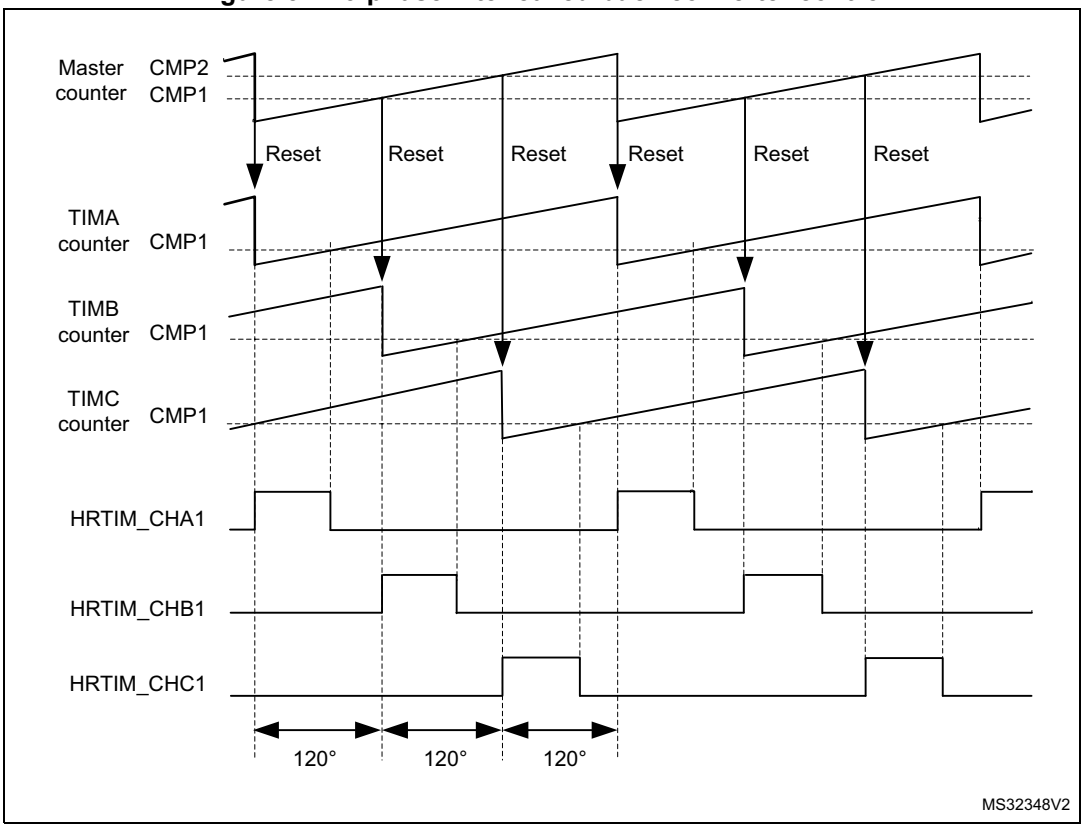
The master timer is responsible for the phase management: it defines the phase relationship between the converters by resetting the timers periodically. The phase-shift is 360° divided by the number of phases, 120° in the given example.

The duty cycle is then programmed into each of the timers. The outputs are defined as follows:

- HRTIM_CHA1 set on master timer period, reset on TACMP1
- HRTIM_CHB1 set on master timer MCMP1, reset on TBCMP1
- HRTIM_CHC1 set on master timer MCMP2, reset on TCCMP1

The ADC trigger can be generated on TxCMP2 compare event. Since all ADC trigger sources are phase-shifted because of the converter topology, it is possible to have all of them combined into a single ADC trigger to save ADC resources (for instance 1 ADC regular channel for the full multiphase converter).

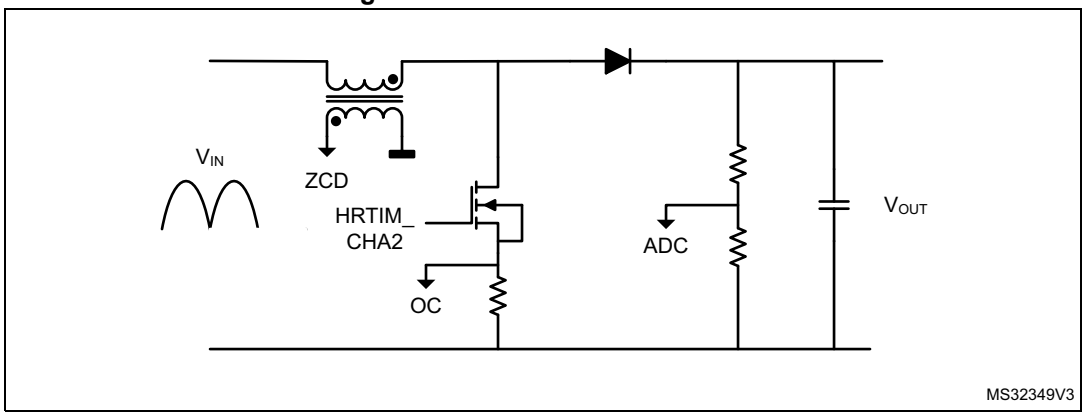
Figure 511. 3-phase interleaved buck converter control



33.4.4 Transition mode power factor correction

The basic operating principle is to build up current into an inductor during a fixed Ton time. This current then decays during the Toff time, and the period is restarted when it becomes null. This is detected using a zero crossing detection circuitry (ZCD), as shown on *Figure 512*. With a constant Ton time, the peak current value in the inductor is proportional to the rectified AC input voltage, which provides the power factor correction.

Figure 512. Transition mode PFC



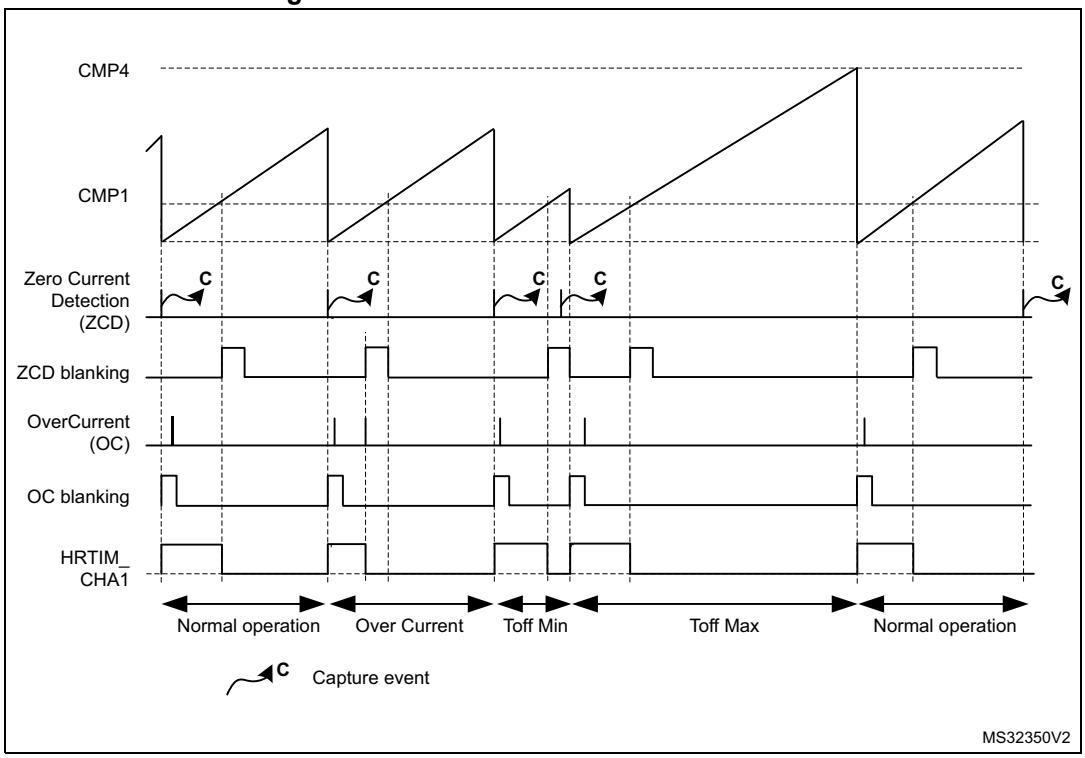
This converter operates with a constant Ton time and a variable frequency due to the Toff time variation (depending on the input voltage). It must also include some features to operate when no zero-crossing is detected, or to limit the Ton time in case of overcurrent (OC). The OC feedback is usually conditioned with the built-in comparator and routed onto an external event input.

Figure 513 presents the waveform during the various operating modes, with the following defined parameters:

- Ton Min: masks spurious overcurrent (freewheeling diode recovery current), represented as OC blanking
- Ton Max: practically, the converter set-point. It is defined by CMP1
- Toff Min: limits the frequency when the current limit is close to zero (demagnetization is very fast). It is defined with CMP2.
- Toff Max: prevents the system to be stuck if no ZCD occurs. It is defined with CMP4 in autodelayed mode.

Both Toff values are autodelayed since the value must be relative to the output falling edge.

Figure 513. Transition mode PFC waveforms



33.5 HRTIM registers

33.5.1 HRTIM master timer control register (HRTIM_MCR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRSTDMA[1:0]		MREPU	Res.	PREEN	DACSYNC[1:0]		Res.	Res.	TFCEN	TECEN	TDCEN	TCCEN	TBCEN	TACEN	MCEN
rw	rw	rw		rw	rw	rw			rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYNCSRC[1:0]		SYNCOUT[1:0]		SYNCSTRTM	SYNCRSTM	SYNCIN[1:0]		INTLVD[1:0]		HALF	RETRIG	CONT	CKPSC[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **BRSTDMA[1:0]**: Burst DMA update

These bits define how the update occurs relatively to a burst DMA transaction.

00: Update done independently from the DMA burst transfer completion

01: Update done when the DMA burst transfer is completed

10: Update done on master timer roll-over following a DMA burst transfer completion. This mode only works in continuous mode.

11: Reserved

Bit 29 **MREPU**: Master timer repetition update

This bit defines whether an update occurs when the master timer repetition period is completed (either due to roll-over or reset events). MREPU can be set only if BRSTDMA[1:0] = 00 or 01.

0: Update on repetition disabled

1: Update on repetition enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **PREEN**: Preload enable

This bit enables the registers preload mechanism and defines whether the write accesses to the memory mapped registers are done into HRTIM's active or preload registers.

0: Preload disabled: the write access is directly done into the active register

1: Preload enabled: the write access is done into the preload register

Bits 26:25 **DACSYNC[1:0]:** DAC synchronization

A DAC synchronization event can be enabled and generated when the master timer update occurs. These bits are defining on which output the DAC synchronization is sent (refer to *Section 33.3.21: DAC triggers* for connections details).

00: No DAC trigger generated

01: Trigger generated on hrtim_dac_trg1

10: Trigger generated on hrtim_dac_trg2

11: Trigger generated on hrtim_dac_trg3

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **TFCEN**: Timer F counter enable

This bit starts the timer F counter.

0: Timer F counter disabled

1: Timer F counter enabled

Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 21 **TECEN**: Timer E counter enable
This bit starts the timer E counter.
0: Timer E counter disabled
1: Timer E counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 20 **TDCEN**: Timer D counter enable
This bit starts the timer D counter.
0: Timer D counter disabled
1: Timer D counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 19 **TCCEN**: Timer C counter enable
This bit starts the timer C counter.
0: Timer C counter disabled
1: Timer C counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 18 **TBCEN**: Timer B counter enable
This bit starts the timer B counter.
0: Timer B counter disabled
1: Timer B counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 17 **TACEN**: Timer A counter enable
This bit starts the timer A counter.
0: Timer A counter disabled
1: Timer A counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 16 **MCEN**: Master timer counter enable
This bit starts the master timer counter.
0: Master counter disabled
1: Master counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bits 15:14 **SYNCSRC[1:0]**: Synchronization source
These bits define the source and event to be sent on the synchronization outputs SYNCOUT[2:1]
00: Master timer start
01: Master timer compare 1 event
10: Timer A start/reset
11: Timer A compare 1 event

Bits 13:12 **SYNCOUT[1:0]**: Synchronization output
These bits define the routing and conditioning of the synchronization output event.
00: Disabled
01: On-chip synchronization sent to other HRTIM instances
10: Positive pulse on HRTIM_SCOUT output (16x f_{HRTIM} clock cycles)
11: Negative pulse on HRTIM_SCOUT output (16x f_{HRTIM} clock cycles)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set)

Bit 11 **SYNCSTRTM**: Synchronization starts master
This bit enables the master timer start when receiving a synchronization input event:
0: No effect on the master timer
1: A synchronization input event starts the master timer

Bit 10 **SYNCRSTM**: Synchronization resets master

This bit enables the master timer reset when receiving a synchronization input event:

0: No effect on the master timer

1: A synchronization input event resets the master timer

Bits 9:8 **SYNCIN[1:0]**: Synchronization input

These bits are defining the synchronization input source.

00: Disabled. HRTIM is not synchronized and runs in standalone mode.

01: Internal event on hrtim_in_sync[1], from other on-chip HRTIM instances

10: Internal event on hrtim_in_sync[2]: the HRTIM is synchronized with the on-chip timer (see *Section : Synchronization input*).

11: External event on hrtim_in_sync[3]: a positive pulse on HRTIM_SCIN input triggers the HRTIM.

Note: This parameter cannot be changed once the impacted timers are enabled.

Bits 7:6 **INTLVD[1:0]**: Interleaved mode

This bitfield is significant only when the HALF bit is reset. It enables the interleaved mode.

00: Interleaved mode disabled

01: Triple interleaved mode: when HRTIM_MPER register is written, the HRTIM_MCMP1R active register is automatically updated with HRTIM_MPER/3 value, and the HRTIM_MCMP2R active register is automatically updated with 2x (HRTIM_MPER/3) value.

10: Quad interleaved mode: when HRTIM_MPER register is written, the HRTIM_MCMP1R active register is automatically updated with HRTIM_MPER/4 value, the HRTIM_MCMP2R active register is automatically updated with HRTIM_MPER/2 value and the HRTIM_MCMP3R active register is automatically updated with 3x (HRTIM_MPER/4) value.

11: Interleaved mode disabled

Bit 5 **HALF**: Half mode

This bit enables the half duty-cycle mode: the HRTIM_MCMP1R active register is automatically updated with HRTIM_MPER/2 value when HRTIM_MPER register is written.

0: Half mode disabled

1: Half mode enabled

Bit 4 **RETRIG**: Re-triggerable mode

This bit defines the behavior of the master timer counter in single-shot mode.

0: The timer is not re-triggerable: a counter reset can be done only if the counter is stopped (period elapsed)

1: The timer is re-triggerable: a counter reset is done whatever the counter state (running or stopped)

Bit 3 **CONT**: Continuous mode

0: The timer operates in single-shot mode and stops when it reaches the MPER value

1: The timer operates in continuous (free-running) mode and rolls over to zero when it reaches the MPER value

Bits 2:0 **CKPSC[2:0]**: Clock prescaler

These bits define the master timer clock prescaler ratio.

The counter clock equivalent frequency ($f_{COUNTER}$) is equal to $f_{HRCK} / 2^{(CKPSC[2:0]-5)}$.

The prescaling ratio cannot be modified once the timer is enabled.

33.5.2 HRTIM master timer interrupt status register (HRTIM_MISR)

Address offset: 0x004

Reset value: 0x0000 0000

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **MUPD**: Master update interrupt flag
This bit is set by hardware when the master timer registers are updated.
0: No master update interrupt occurred
1: Master update interrupt occurred

Bit 5 **SYNC**: Sync input interrupt flag
This bit is set by hardware when a synchronization input event is received.
0: No sync input interrupt occurred
1: Sync input interrupt occurred

Bit 4 **MREP**: Master repetition interrupt flag
This bit is set by hardware when the master timer repetition period has elapsed.
0: No master repetition interrupt occurred
1: Master repetition interrupt occurred

Bit 3 **MCMP4**: Master compare 4 interrupt flag
Refer to MCMP1 description

Bit 2 **MCMP3**: Master compare 3 interrupt flag
Refer to MCMP1 description

Bit 1 **MCMP2**: Master compare 2 interrupt flag
Refer to MCMP1 description

Bit 0 **MCMP1**: Master compare 1 interrupt flag
This bit is set by hardware when the master timer counter matches the value programmed in the master compare 1 register.
0: No master compare 1 interrupt occurred
1: Master compare 1 interrupt occurred

33.5.3 HRTIM master timer interrupt clear register (HRTIM_MICR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDC	SYNCC	MREPC	MCMP4C	MCMP3C	MCMP2C	MCMP1C
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **MUPDC**: Master update interrupt flag clear
Writing 1 to this bit clears the MUPDC flag in HRTIM_MISR register.

Bit 5 **SYNCC**: Sync input interrupt flag clear
Writing 1 to this bit clears the SYNC flag in HRTIM_MISR register.

Bit 4 **MREPC**: Repetition interrupt flag clear
Writing 1 to this bit clears the MREP flag in HRTIM_MISR register.

Bit 3 **MCMP4C**: Master compare 4 interrupt flag clear
Writing 1 to this bit clears the MCMP4 flag in HRTIM_MISR register.

Bit 2 **MCMP3C**: Master compare 3 interrupt flag clear
Writing 1 to this bit clears the MCMP3 flag in HRTIM_MISR register.

Bit 1 **MCMP2C**: Master compare 2 interrupt flag clear
Writing 1 to this bit clears the MCMP2 flag in HRTIM_MISR register.

Bit 0 **MCMP1C**: Master compare 1 interrupt flag clear
Writing 1 to this bit clears the MCMP1 flag in HRTIM_MISR register.

33.5.4 HRTIM master timer DMA interrupt enable register (HRTIM_MDIER)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDDE	SYNCDE	MREPDE	MCMP4DE	MCMP3DE	MCMP2DE	MCMP1DE
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDIE	SYNCIE	MREPIE	MCMP4IE	MCMP3IE	MCMP2IE	MCMP1IE
									rw	rw	rw	rw	rw	rw	rw

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

Bit 22 **MUPDDE**: Master update DMA request enable
This bit is set and cleared by software to enable/disable the master update DMA requests.
0: Master update DMA request disabled
1: Master update DMA request enabled

Bit 21 **SYNCDE**: Sync input DMA request enable
This bit is set and cleared by software to enable/disable the sync input DMA requests.
0: Sync input DMA request disabled
1: Sync input DMA request enabled

Bit 20 **MREPDE**: Master repetition DMA request enable
This bit is set and cleared by software to enable/disable the master timer repetition DMA requests.
0: Repetition DMA request disabled
1: Repetition DMA request enabled

Bit 19 **MCMP4DE**: Master compare 4 DMA request enable
Refer to MCMP1DE description

Bit 18 **MCMP3DE**: Master compare 3 DMA request enable
Refer to MCMP1DE description

Bit 17 **MCMP2DE**: Master compare 2 DMA request enable
Refer to MCMP1DE description

Bit 16 **MCMP1DE**: Master compare 1 DMA request enable
This bit is set and cleared by software to enable/disable the master timer compare 1 DMA requests.
0: Compare 1 DMA request disabled
1: Compare 1 DMA request enabled

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

Bit 6 **MUPDIE**: Master update interrupt enable
This bit is set and cleared by software to enable/disable the master timer registers update interrupts
0: Master update interrupts disabled
1: Master update interrupts enabled

Bit 5 **SYNCIE**: Sync input interrupt enable
This bit is set and cleared by software to enable/disable the sync input interrupts
0: Sync input interrupts disabled
1: Sync input interrupts enabled

Bit 4 **MREPIE**: Master repetition interrupt enable
This bit is set and cleared by software to enable/disable the master timer repetition interrupts
0: Master repetition interrupt disabled
1: Master repetition interrupt enabled

Bit 3 **MCMP4IE**: Master compare 4 interrupt enable
Refer to MCMP1IE description

Bit 2 **MCMP3IE**: Master compare 3 interrupt enable
Refer to MCMP1IE description

Bit 1 **MCMP2IE**: Master compare 2 interrupt enable
Refer to MCMP1IE description

Bit 0 **MCMP1IE**: Master compare 1 interrupt enable
This bit is set and cleared by software to enable/disable the master timer compare 1 interrupt
0: Compare 1 interrupt disabled
1: Compare 1 interrupt enabled

33.5.5 HRTIM master timer counter register (HRTIM_MCNTR)

Address offset: 0x010

Reset value: 0x0000 0000

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCNT[15:0]**: Counter value

Holds the master timer counter value. This register can only be written when the master timer is stopped (MCEN = 0 in HRTIM_MCR).

Note: The timer behavior is not guaranteed if the counter value is set above the HRTIM_MPER register value.

33.5.6 HRTIM master timer period register (HRTIM_MPER)

Address offset: 0x014

Reset value: 0x0000 FFDF

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MPER[15:0]**: Master timer period value

This register defines the counter overflow value.

The period value must be above or equal to 3 periods of the f_{HRTIM} clock.

The maximum value is 0x0000 FFDF.

33.5.7 HRTIM master timer repetition register (HRTIM_MREP)

Address offset: 0x018

Reset value: 0x0000 0000

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

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

Bits 7:0 **MREP[7:0]**: Master timer repetition period value

This register holds the repetition period value for the master counter. It is either the preload register or the active register if preload is disabled.

33.5.8 HRTIM master timer compare 1 register (HRTIM_MCMP1R)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCMP1[15:0]**: Master timer compare 1 value

This register holds the master timer compare 1 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock.

33.5.9 HRTIM master timer compare 2 register (HRTIM_MCMP2R)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCMP2[15:0]**: Master timer compare 2 value

This register holds the master timer compare 2 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock.

33.5.10 HRTIM master timer compare 3 register (HRTIM_MCMP3R)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCMP3[15:0]**: Master timer compare 3 value

This register holds the master timer compare 3 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock.

33.5.11 HRTIM master timer compare 4 register (HRTIM_MCMP4R)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCMP4[15:0]**: Master timer compare 4 value

This register holds the master timer compare 4 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock.

33.5.12 HRTIM timer x control register (HRTIM_TIMxCR) (x = A to F)

Address offset: Block A: 0x100

Address offset: Block B: 0x200

Address offset: Block C: 0x300

Address offset: Block D: 0x400

Address offset: Block E: 0x500

Address offset: Block F: 0x600

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDGAT[3:0]				PREEN	DACSYNC[1:0]		MSTU	TEU	TDU	TCU	TBU	TAU	TxRSTU	TxREPU	TFU
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DELCMP4[1:0]		DELCMP2[1:0]		SYNCSTRTx	SYNCRSTx	RSYNCU	INTLVD[1:0]		PSHPLL	HALF	RETRIG	CONT	CKPSCx[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **UPDGAT[3:0]**: Update gating

These bits define how the update occurs relatively to the burst DMA transaction and the external update request on update enable inputs hrtim_upd_en[3:1] (see *Table 327*).

The update events, as mentioned below, can be: MSTU, TFU, TEU, TDU, TCU, TBU, TAU, TxRSTU, TxREPU.

0000: The update occurs independently from the DMA burst transfer

0001: The update occurs when the DMA burst transfer is completed

0010: The update occurs on the update event following the DMA burst transfer completion

0011: The update occurs on a rising edge on hrtim_upd_en1

0100: The update occurs on a rising edge on hrtim_upd_en2

0101: The update occurs on a rising edge on hrtim_upd_en3

0110: The update occurs on the update event following a rising edge on hrtim_upd_en1

0111: The update occurs on the update event following a rising edge on hrtim_upd_en2

1000: The update occurs on the update event following a rising edge on hrtim_upd_en3

Others: Reserved

Note: This bitfield must be reset before programming a new value.

For UPDGAT[3:0] values equal to 0001, 0011, 0100, 0101, it is possible to have multiple concurrent update source (for instance RSTU and DMA burst).

Bit 27 **PREEN**: Preload enable

This bit enables the registers preload mechanism and defines whether a write access into a preload-able register is done into the active or the preload register.

0: Preload disabled: the write access is directly done into the active register

1: Preload enabled: the write access is done into the preload register

Bits 26:25 **DACSYNC[1:0]**: DAC synchronization

A DAC synchronization event is generated when the timer update occurs. These bits are defining on which output the DAC synchronization is sent (refer to *Section 33.3.21: DAC triggers* for connections details).

00: No DAC trigger generated
01: Trigger generated on hrtim_dac_trg1
10: Trigger generated on hrtim_dac_trg2
11: Trigger generated on hrtim_dac_trg3

Bit 24 **MSTU**: Master timer update

Register update is triggered by the master timer update.
0: Update by master timer disabled
1: Update by master timer enabled

Bit 23 **TEU**: Timer E update

Register update is triggered by the timer E update
0: Update by timer E disabled
1: Update by timer E enabled

Note: This bit is reserved for HRTIM_TIMECR. *It is only available for HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMFCR.*

Bit 22 **TDU**: Timer D update

Register update is triggered by the timer D update
0: Update by timer D disabled
1: Update by timer D enabled

Note: This bit is reserved for HRTIM_TIMDCR. It is only available for HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMECR, HRTIM_TIMFCR.

Bit 21 **TCU**: Timer C update

Register update is triggered by the timer C update
0: Update by timer C disabled
1: Update by timer C enabled

Note: This bit is reserved for HRTIM_TIMCCR. It is only available for HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMDCR, HRTIM_TIMECR, HRTIM_TIMFCR.

Bit 20 **TBU**: Timer B update

Register update is triggered by the timer B update
0: Update by timer B disabled
1: Update by timer B enabled

Note: This bit is reserved for HRTIM_TIMBCR. It is only available for HRTIM_TIMACR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMECR, HRTIM_TIMFCR.

Bit 19 **TAU**: Timer A update

Register update is triggered by the timer A update
0: Update by timer A disabled
1: Update by timer A enabled

Note: This bit is reserved for HRTIM_TIMBCR. It is only available for HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMECR, HRTIM_TIMFCR.

Bit 18 **TxRSTU**: Timer x reset update

Register update is triggered by timer x counter reset or roll-over to 0 after reaching the period value in continuous mode.
0: Update by timer x reset / roll-over disabled
1: Update by timer x reset / roll-over enabled

Bit 17 **TxREPU**: Timer x repetition update

Register update is triggered when the counter rolls over and HRTIM_REPx = 0

0: Update on repetition disabled

1: Update on repetition enabled

Bit 16 **TFU**: Timer F update

Register update is triggered by the timer F update

0: Update by timer F disabled

1: Update by timer F enabled

Note: This bit is reserved for HRTIM_TIMFCR. It is only available for HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMECR.

Bits 15:14 **DELCMP4[1:0]**: CMP4 auto-delayed mode

This bitfield defines whether the compare register is behaving in standard mode (compare match issued as soon as counter equal compare), or in auto-delayed mode (see *Section : Autodelayed mode*).

00: CMP4 register is always active (standard compare mode)

01: CMP4 value is recomputed and is active following a capture 2 event

10: CMP4 value is recomputed and is active following a capture 2 event, or is recomputed and active after Compare 1 match (timeout function if capture 2 event is missing)

11: CMP4 value is recomputed and is active following a capture 2 event, or is recomputed and active after Compare 3 match (timeout function if capture event is missing)

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bits 13:12 **DELCMP2[1:0]**: CMP2 auto-delayed mode

This bitfield defines whether the compare register is behaving in standard mode (compare match issued as soon as counter equal compare), or in auto-delayed mode (see *Section : Autodelayed mode*).

00: CMP2 register is always active (standard compare mode)

01: CMP2 value is recomputed and is active following a capture 1 event

10: CMP2 value is recomputed and is active following a capture 1 event, or is recomputed and active after compare 1 match (timeout function if capture event is missing)

11: CMP2 value is recomputed and is active following a capture 1 event, or is recomputed and active after compare 3 match (timeout function if capture event is missing)

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 11 **SYNCSTRTx**: Synchronization starts timer x

This bit defines the timer x behavior following the synchronization event:

0: No effect on timer x

1: A synchronization input event starts the timer x

Bit 10 **SYNCRSTx**: Synchronization resets timer x

This bit defines the timer x behavior following the synchronization event:

0: No effect on timer x

1: A synchronization input event resets the timer x

Bit 9 **RSYNCU**: Re-synchronized update

This bit specifies whether update source coming outside from the timing unit must be synchronized:

0: The update coming from adjacent timers (when MSTU, TAU, TBU, TCU, TDU, TEU, TFU bit is set) or from a software update (TxSWU bit) is taken into account immediately

1: The update coming from adjacent timers (when MSTU, TAU, TBU, TCU, TDU, TEU, TFU bit is set) or from a software update (TxSWU bit) is taken into account on the following reset/roll-over event.

Note: This bit is significant only when UPDGAT[3:0] = 0000, it is ignored otherwise.

Bits 8:7 **INTLVD[1:0]**: Interleaved mode

This bitfield is significant only when the HALF bit is reset. It enables the interleaved mode.

00: Interleaved mode disabled

01: Triple interleaved mode: when HRTIM_PERxR register is written, the HRTIM_CMP1xR active register is automatically updated with HRTIM_PERxR/3 value, and the HRTIM_CMP2xR active register is automatically updated with 2x (HRTIM_PERxR/3) value.

10: Quad interleaved mode: when HRTIM_PERxR register is written, the HRTIM_CMP1xR active register is automatically updated with HRTIM_PERxR/4 value, the HRTIM_CMP2xR active register is automatically updated with HRTIM_PERxR/2 value and the HRTIM_CMP3xR active register is automatically updated with 3x (HRTIM_PERxR/4) value.

11: Interleaved mode disabled

Bit 6 **PSHPLL**: Push-pull mode enable

This bit enables the push-pull mode.

0: Push-pull mode disabled

1: Push-pull mode enabled

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 5 **HALF**: Half mode enable

This bit enables the half duty-cycle mode: the HRTIM_CMP1xR active register is automatically updated with HRTIM_PERxR/2 value when HRTIM_PERxR register is written.

0: Half mode disabled

1: Half mode enabled

Bit 4 **RETRIG**: Re-triggerable mode

This bit defines the counter behavior in single shot mode.

0: The timer is not re-triggerable: a counter reset is done if the counter is stopped (period elapsed in single-shot mode or counter stopped in continuous mode)

1: The timer is re-triggerable: a counter reset is done whatever the counter state.

Bit 3 **CONT**: Continuous mode

This bit defines the timer operating mode.

0: The timer operates in single-shot mode and stops when it reaches TIMxPER value

1: The timer operates in continuous mode and rolls over to zero when it reaches TIMxPER value

Bits 2:0 **CKPSCx[2:0]**: HRTIM timer x clock prescaler

These bits define the master timer clock prescaler ratio.

The counter clock equivalent frequency ($f_{COUNTER}$) is equal to $f_{HRCK} / 2^{(CKPSC[2:0]-5)}$.

The prescaling ratio cannot be modified once the timer is enabled.

33.5.13 HRTIM timer x interrupt status register (HRTIM_TIMxISR) (x = A to F)

Address offset: Block A: 0x104

Address offset: Block B: 0x204

Address offset: Block C: 0x304

Address offset: Block D: 0x404

Address offset: Block E: 0x504

Address offset: Block F: 0x604

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COTS	COS	O2CPY	O1CPY	O2STAT	O1STAT	IPPSTAT	CPPSTAT
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CORDE	DLYPRT	RST	RSTx2	SETx2	RSTx1	SETx1	CPT2	CPT1	UPD	COT	REP	CMP4	CMP3	CMP2	CMP1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **COTS**: Crossbar Override Trigger status

This status bit is a raw copy of the crossbar override trigger input. It can be polled to verify if the crossbar override mode can be rearmed

0: Crossbar Override trigger is inactive

1: Crossbar Override trigger is active

Bit 22 **COS**: Crossbar Override status

This status bit indicates the crossbar override status.

0: Crossbar Override is inactive (crossbar is active)

1: Crossbar Override is active

Bit 21 **O2CPY**: Output 2 copy

This status bit is a raw copy of the output 2 state, before the output stage (chopper, polarity). It allows to check the current output state before re-enabling the output after a delayed protection.

0: Output 2 is inactive

1: Output 2 is active

Bit 20 **O1CPY**: Output 1 copy

This status bit is a raw copy of the output 1 state, before the output stage (chopper, polarity). It allows to check the current output state before re-enabling the output after a delayed protection.

0: Output 1 is inactive

1: Output 1 is active

Bit 19 **O2STAT**: Output 2 status

This status bit indicates the output 2 state when the delayed idle protection was triggered. This bit is updated upon any new delayed protection entry. This bit is not updated in balanced idle.

0: Output 2 was inactive

1: Output 2 was active

Bit 18 **O1STAT**: Output 1 status

This status bit indicates the output 1 state when the delayed idle protection was triggered. This bit is updated upon any new delayed protection entry. This bit is not updated in balanced idle.

0: Output 1 was inactive

1: Output 1 was active

Bit 17 **IPPSTAT**: Idle push-pull Status

This status bit indicates on which output the signal was applied, in push-pull mode balanced fault mode or delayed idle mode, when the protection was triggered (whatever the output state, active or inactive).

0: Protection occurred when the output 1 was active and output 2 forced inactive

1: Protection occurred when the output 2 was active and output 1 forced inactive

Bit 16 **CPPSTAT**: Current push-pull status

This status bit indicates on which output the signal is currently applied, in push-pull mode. It is only significant in this configuration.

0: Signal applied on output 1 and output 2 forced inactive

1: Signal applied on output 2 and output 1 forced inactive

Bit 15 **CORDE**: Crossbar Override Rearm Delay Elapsed flag

This bit is set when the crossbar override mode can be rearmed. This interrupt flag is asserted when at least on of the one of the 2 channels is configured to Delayed fault force modes (COFMx[2:0] = 000 and 001).

0: No Crossbar override rearm delay elapsed interrupt occurred

1: Crossbar override rearm delay elapsed interrupt occurred

Bit 14 **DLYPRT**: Delayed protection flag

0: No delayed protection interrupt occured

1: Delayed Idle or balanced Idle mode entry occured

Bit 13 **RST**: Reset and/or roll-over interrupt flag

This bit is set by hardware when the timer x counter is reset or rolls over in continuous mode.

0: No TIMx counter reset/roll-over interrupt occurred

1: TIMX counter reset/roll-over interrupt occurred

Bit 12 **RSTx2**: Output 2 reset interrupt flag timer x

Refer to RSTx1 description

Bit 11 **SETx2**: Output 2 set interrupt flag timer x

Refer to SETx1 description

Bit 10 **RSTx1**: Output 1 reset interrupt flag timer x

This bit is set by hardware when the Tx1 output is reset (goes from active to inactive mode).

0: No Tx1 output reset interrupt occurred

1: Tx1 output reset interrupt occurred

Bit 9 **SETx1**: Output 1 set interrupt flag timer x

This bit is set by hardware when the Tx1 output is set (goes from inactive to active mode).

0: No Tx1 output set interrupt occurred

1: Tx1 output set interrupt occurred

Bit 8 **CPT2**: Capture 2 interrupt flag

Refer to CPT1 description

Bit 7 **CPT1**: Capture 1 interrupt flag

This bit is set by hardware when the timer x capture 1 event occurs.

0: No timer x capture 1 interrupt occurred

1: Timer x capture 1 interrupt occurred

Bit 6 **UPD**: Update interrupt flag
This bit is set by hardware when the timer x update event occurs.
0: No timer x update interrupt occurred
1: Timer x update interrupt occurred

Bit 5 **COT**: Crossbar Override Trigger flag
This bit is set when the crossbar override mode is triggered.
0: No Crossbar override trigger interrupt occurred
1: Crossbar override trigger interrupt occured

Bit 4 **REP**: Repetition interrupt flag
This bit is set by hardware when the timer x repetition period has elapsed.
0: No timer x repetition interrupt occurred
1: Timer x repetition interrupt occurred

Bit 3 **CMP4**: Compare 4 interrupt flag
Refer to CMP1 description

Bit 2 **CMP3**: Compare 3 interrupt flag
Refer to CMP1 description

Bit 1 **CMP2**: Compare 2 interrupt flag
Refer to CMP1 description

Bit 0 **CMP1**: Compare 1 interrupt flag
This bit is set by hardware when the timer x counter matches the value programmed in the compare 1 register.
0: No compare 1 interrupt occurred
1: Compare 1 interrupt occurred

33.5.14 HRTIM timer x interrupt clear register (HRTIM_TIMxICR) (x = A to F)

Address offset: Block A: 0x108

Address offset: Block B: 0x208

Address offset: Block C: 0x308

Address offset: Block D: 0x408

Address offset: Block E: 0x508

Address offset: Block F: 0x608

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CORDEC	DLYPRTC	RSTC	RSTx2C	SETx2C	RSTx1C	SETx1C	CPT2C	CPT1C	UPDC	COTC	REPC	CMP4C	CMP3C	CMP2C	CMP1C
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CORDEC**: Crossbar Override Rearm Delay Elapsed flag clear
Writing 1 to this bit clears the CORDE flag in HRTIM_TIMxISR register

Bit 14 **DLYPRTC**: Delayed protection flag clear
Writing 1 to this bit clears the DLYPRT flag in HRTIM_TIMxISR register

Bit 13 **RSTC**: Reset interrupt flag clear
Writing 1 to this bit clears the RST flag in HRTIM_TIMxISR register

Bit 12 **RSTx2C**: Output 2 reset flag clear timer x
Writing 1 to this bit clears the RSTx2 flag in HRTIM_TIMxISR register

Bit 11 **SETx2C**: Output 2 set flag clear timer x
Writing 1 to this bit clears the SETx2 flag in HRTIM_TIMxISR register

Bit 10 **RSTx1C**: Output 1 reset flag clear timer x
Writing 1 to this bit clears the RSTx1 flag in HRTIM_TIMxISR register

Bit 9 **SETx1C**: Output 1 set flag clear timer x
Writing 1 to this bit clears the SETx1 flag in HRTIM_TIMxISR register

Bit 8 **CPT2C**: Capture 2 interrupt flag clear
Writing 1 to this bit clears the CPT2 flag in HRTIM_TIMxISR register

Bit 7 **CPT1C**: Capture 1 interrupt flag clear
Writing 1 to this bit clears the CPT1 flag in HRTIM_TIMxISR register

Bit 6 **UPDC**: Update interrupt flag clear
Writing 1 to this bit clears the UPD flag in HRTIM_TIMxISR register

Bit 5 **COTC**: Crossbar Override Trigger flag clear
Writing 1 to this bit clears the COT flag in HRTIM_TIMxISR register

Bit 4 **REPC**: Repetition interrupt flag clear
Writing 1 to this bit clears the REP flag in HRTIM_TIMxISR register

Bit 3 **CMP4C**: Compare 4 interrupt flag clear
Writing 1 to this bit clears the CMP4 flag in HRTIM_TIMxISR register

Bit 2 **CMP3C**: Compare 3 interrupt flag clear
Writing 1 to this bit clears the CMP3 flag in HRTIM_TIMxISR register

Bit 1 **CMP2C**: Compare 2 interrupt flag clear
Writing 1 to this bit clears the CMP2 flag in HRTIM_TIMxISR register

Bit 0 **CMP1C**: Compare 1 interrupt flag clear
Writing 1 to this bit clears the CMP1 flag in HRTIM_TIMxISR register

33.5.15 HRTIM timer x DMA interrupt enable register (HRTIM_TIMxDIER) (x = A to F)

Address offset: Block A: 0x10C

Address offset: Block B: 0x20C

Address offset: Block C: 0x30C

Address offset: Block D: 0x40C

Address offset: Block E: 0x50C

Address offset: Block F: 0x60C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CORDEDE	DLYPRTDE	RSTDE	RSTx2DE	SETx2DE	RSTx1DE	SETx1DE	CPT2DE	CPT1DE	UPDDE	COTDE	REPDE	CMP4DE	CMP3DE	CMP2DE	CMP1DE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CORDEIE	DLYPRTIE	RSTIE	RSTx2IE	SETx2IE	RSTx1IE	SETx1IE	CPT2IE	CPT1IE	UPDIE	COTIE	REPIE	CMP4IE	CMP3IE	CMP2IE	CMP1IE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CORDEDE**: Crossbar override rearm delay elapsed DMA request enable
This bit is set and cleared by software to enable/disable DMA requests on Crossbar override rearm delay elapsed event.
0: Crossbar override rearm delay elapsed DMA request disabled
1: Crossbar override rearm delay elapsed DMA request enabled

Bit 30 **DLYPRTDE**: Delayed protection DMA request enable
This bit is set and cleared by software to enable/disable DMA requests on delayed protection.
0: Delayed protection DMA request disabled
1: Delayed protection DMA request enabled

Bit 29 **RSTDE**: Reset/roll-over DMA request enable
This bit is set and cleared by software to enable/disable DMA requests on timer x counter reset or roll-over in continuous mode.
0: Timer x counter reset/roll-over DMA request disabled
1: Timer x counter reset/roll-over DMA request enabled

Bit 28 **RSTx2DE**: Output 2 reset DMA request enable timer x
Refer to RSTx1DE description

Bit 27 **SETx2DE**: Output 2 set DMA request enable timer x
Refer to SETx1DE description

Bit 26 **RSTx1DE**: Output 1 reset DMA request enable timer x
This bit is set and cleared by software to enable/disable Tx1 output reset DMA requests.
0: Tx1 output reset DMA request disabled
1: Tx1 output reset DMA request enabled

Bit 25 **SETx1DE**: Output 1 set DMA request enable timer x
This bit is set and cleared by software to enable/disable Tx1 output set DMA requests.
0: Tx1 output set DMA request disabled
1: Tx1 output set DMA request enabled

Bit 24 **CPT2DE**: Capture 2 DMA request enable
Refer to CPT1DE description

Bit 23 **CPT1DE**: Capture 1 DMA request enable
This bit is set and cleared by software to enable/disable capture 1 DMA requests.
0: Capture 1 DMA request disabled
1: Capture 1 DMA request enabled

Bit 22 **UPDDE**: Update DMA request enable
This bit is set and cleared by software to enable/disable DMA requests on update event.
0: Update DMA request disabled
1: Update DMA request enabled

Bit 21 **COTDE**: Crossbar Override Trigger DMA request enable
This bit is set and cleared by software to enable/disable DMA requests on Crossbar Override Trigger event.
0: Crossbar Override Trigger DMA request disabled
1: Crossbar Override Trigger DMA request enabled

Bit 20 **REPDE**: Repetition DMA request enable
This bit is set and cleared by software to enable/disable DMA requests on repetition event.
0: Repetition DMA request disabled
1: Repetition DMA request enabled

Bit 19 **CMP4DE**: Compare 4 DMA request enable
Refer to CMP1DE description

Bit 18 **CMP3DE**: Compare 3 DMA request enable
Refer to CMP1DE description

Bit 17 **CMP2DE**: Compare 2 DMA request enable
Refer to CMP1DE description

Bit 16 **CMP1DE**: Compare 1 DMA request enable
This bit is set and cleared by software to enable/disable the compare 1 DMA requests.
0: Compare 1 DMA request disabled
1: Compare 1 DMA request enabled

Bit 15 **CORDEIE**: Crossbar override rearm delay elapsed interrupt enable
This bit is set and cleared by software to enable/disable interrupts on Crossbar override rearm delay elapsed event.
0: Crossbar override rearm delay elapsed interrupts disabled
1: Crossbar override rearm delay elapsed interrupts enabled

Bit 14 **DLYPRTIE**: Delayed protection interrupt enable
This bit is set and cleared by software to enable/disable interrupts on delayed protection.
0: Delayed protection interrupts disabled
1: Delayed protection interrupts enabled

Bit 13 **RSTIE**: Reset/roll-over interrupt enable
This bit is set and cleared by software to enable/disable interrupts on timer x counter reset or roll-over in continuous mode.
0: Timer x counter reset/roll-over interrupt disabled
1: Timer x counter reset/roll-over interrupt enabled

Bit 12 **RSTx2IE**: Output 2 reset interrupt enable timer x
Refer to RSTx1IE description

Bit 11 **SETx2IE**: Output 2 set interrupt enable timer x
Refer to SETx1IE description

Bit 10 **RSTx1IE**: Output 1 reset interrupt enable timer x
This bit is set and cleared by software to enable/disable Tx1 output reset interrupts.
0: Tx1 output reset interrupts disabled
1: Tx1 output reset interrupts enabled

Bit 9 **SETx1IE**: Output 1 set interrupt enable timer x
This bit is set and cleared by software to enable/disable Tx1 output set interrupts.
0: Tx1 output set interrupts disabled
1: Tx1 output set interrupts enabled

Bit 8 **CPT2IE**: Capture interrupt enable
Refer to CPT1IE description

Bit 7 **CPT1IE**: Capture interrupt enable
This bit is set and cleared by software to enable/disable capture 1 interrupts.
0: Capture 1 interrupts disabled
1: Capture 1 interrupts enabled

Bit 6 **UPDIE**: Update interrupt enable
This bit is set and cleared by software to enable/disable update event interrupts.
0: Update interrupts disabled
1: Update interrupts enabled

Bit 5 **COTIE**: Crossbar Override Trigger interrupt enable
This bit is set and cleared by software to enable/disable Crossbar Override Trigger interrupts.
0: Crossbar Override Trigger interrupts disabled
1: Crossbar Override Trigger enabled

Bit 4 **REPIE**: Repetition interrupt enable
This bit is set and cleared by software to enable/disable repetition event interrupts.
0: Repetition interrupts disabled
1: Repetition interrupts enabled

Bit 3 **CMP4IE**: Compare 4 interrupt enable
Refer to CMP1IE description

Bit 2 **CMP3IE**: Compare 3 interrupt enable
Refer to CMP1IE description

Bit 1 **CMP2IE**: Compare 2 interrupt enable
Refer to CMP1IE description

Bit 0 **CMP1IE**: Compare 1 interrupt enable
This bit is set and cleared by software to enable/disable the compare 1 interrupts.
0: Compare 1 interrupt disabled
1: Compare 1 interrupt enabled

33.5.16 HRTIM timer x counter register (HRTIM_CNTxR) (x = A to F)

Address offset: Block A: 0x110

Address offset: Block B: 0x210

Address offset: Block C: 0x310

Address offset: Block D: 0x410

Address offset: Block E: 0x510

Address offset: Block F: 0x610

Reset value: 0x0000 0000

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CNTx[15:0]**: Timer x counter value

This register holds the timer x counter value. It can only be written when the timer is stopped (TxCEN = 0 in HRTIM_TIMxCR).

Note: The timer behavior is not guaranteed if the counter value is above the HRTIM_PERxR register value.

33.5.17 HRTIM timer x period register (HRTIM_PERxR) (x = A to F)

Address offset: Block A: 0x114

Address offset: Block B: 0x214

Address offset: Block C: 0x314

Address offset: Block D: 0x414

Address offset: Block E: 0x514

Address offset: Block F: 0x614

Reset value: 0x0000 FFDF

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PERx[15:0]**: Timer x period value

This register holds timer x period value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The period value must be above or equal to 3 periods of the f_{HRTIM} clock.

The maximum value is 0x0000 FFDF.

33.5.18 HRTIM timer x repetition register (HRTIM_REPxR) (x = A to F)

Address offset: Block A: 0x118

Address offset: Block B: 0x218

Address offset: Block C: 0x318

Address offset: Block D: 0x418

Address offset: Block E: 0x518

Address offset: Block F: 0x618

Reset value: 0x0000 0000

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

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

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

Bits 7:0 **REPx[7:0]**: Timer x repetition period value

This register holds the repetition period value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

33.5.19 HRTIM timer x compare 1 register (HRTIM_CMP1xR) (x = A to F)

Address offset: Block A: 0x11C

Address offset: Block B: 0x21C

Address offset: Block C: 0x31C

Address offset: Block D: 0x41C

Address offset: Block E: 0x51C

Address offset: Block F: 0x61C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP1x[15:0]**: Timer x compare 1 value

This register holds the compare 1 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be either null or above or equal to 3 periods of the f_{HRTIM} clock.

The null value is programmed following the use case described in *Section : Null duty cycle exception case*.

33.5.20 HRTIM timer x compare 1 compound register (HRTIM_CMP1CxR) (x = A to F)

Address offset: Block A: 0x120

Address offset: Block B: 0x220

Address offset: Block C: 0x320

Address offset: Block D: 0x420

Address offset: Block E: 0x520

Address offset: Block F: 0x620

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **REPx[7:0]**: Timer x repetition value aliased from HRTIM_REPx register
This bitfield is an alias from the REPx[7:0] bitfield in the HRTIMx_REPxR register.

Bits 15:0 **CMP1x[15:0]**: Timer x compare 1 value
This bitfield is an alias from the CMP1x[15:0] bitfield in the HRTIMx_CMP1xR register.

33.5.21 HRTIM timer x compare 2 register (HRTIM_CMP2xR) (x = A to F)

Address offset: Block A: 0x124

Address offset: Block B: 0x224

Address offset: Block C: 0x324

Address offset: Block D: 0x424

Address offset: Block E: 0x524

Address offset: Block F: 0x624

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP2x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP2x[15:0]**: Timer x compare 2 value

This register holds the compare 2 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be either null or above or equal to 3 periods of the f_{HRTIM} clock.

The null value is programmed following the use case described in *Section : Null duty cycle exception case*.

This register behaves as an auto-delayed compare register, if enabled with DELCMP2[1:0] bits in HRTIM_TIMxCR. In this case, the compare value must be strictly above or equal to 3 periods of the f_{HRTIM} clock.

33.5.22 HRTIM timer x compare 3 register (HRTIM_CMP3xR) (x = A to F)

Address offset: Block A: 0x128

Address offset: Block B: 0x228

Address offset: Block C: 0x328

Address offset: Block D: 0x428

Address offset: Block E: 0x528

Address offset: Block F: 0x628

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP3x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP3x[15:0]**: Timer x compare 3 value

This register holds the compare 3 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be either null or above or equal to 3 periods of the f_{HRTIM} clock.

The null value is programmed following the use case described in *Section : Null duty cycle exception case*.

33.5.23 HRTIM timer x compare 4 register (HRTIM_CMP4xR) (x = A to F)

Address offset: Block A: 0x12C

Address offset: Block B: 0x22C

Address offset: Block C: 0x32C

Address offset: Block D: 0x42C

Address offset: Block E: 0x52C

Address offset: Block F: 0x62C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP4x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP4x[15:0]**: Timer x compare 4 value

This register holds the compare 4 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be either null or above or equal to 3 periods of the f_{HRTIM} clock.

The null value is programmed following the use case described in *Section : Null duty cycle exception case*.

This register can behave as an auto-delayed compare register, if enabled with DELCMP4[1:0] bits in HRTIM_TIMxCR. In this case, the compare value must be strictly above or equal to 3 periods of the f_{HRTIM} clock.

33.5.24 HRTIM timer x capture 1 register (HRTIM_CPT1xR) (x = A to F)

Address offset: Block A: 0x130

Address offset: Block B: 0x230

Address offset: Block C: 0x330

Address offset: Block D: 0x430

Address offset: Block E: 0x530

Address offset: Block F: 0x630

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPT1x[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DIR**: Timer x capture 1 direction status
This register holds the counting direction value when the capture 1 event occurred:
0: timer is up-counting
1: timer is down-counting
In up-counting mode (UDM bit reset), the DIR bit is always read as 0.

Bits 15:0 **CPT1x[15:0]**: Timer x capture 1 value
This register holds the counter value when the capture 1 event occurred.

Note: In up/down mode (UDM bit set to 1), the capture value is referred to:
- counting reset, when up-counting
- the PER event when down counting
The DIR bit is used to discriminate the up-down phases when reading the captured value.

33.5.25 HRTIM timer x capture 2 register (HRTIM_CPT2xR) (x = A to F)

Address offset: Block A: 0x134

Address offset: Block B: 0x234

Address offset: Block C: 0x334

Address offset: Block D: 0x434

Address offset: Block E: 0x534

Address offset: Block F: 0x634

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIR
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPT2x[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DIR**: Timer x capture 1 direction status
This register holds the counting direction value when the capture 1 event occurred:
0: timer is up-counting
1: timer is down-counting
In up-counting mode (UDM bit reset), the DIR bit is always read as 0.

Bits 15:0 **CPT2x[15:0]**: Timer x capture 2 value
This register holds the counter value when the capture 2 event occurred.

Note: In up/down mode (UDM bit set to 1), the capture value is referred to:
- counting reset, when up-counting
- the PER event when down counting

The DIR bit is used to discriminate the up-down phases when reading the captured value.

33.5.26 HRTIM timer x deadtime register (HRTIM_DTxR) (x = A to F)

Address offset: Block A: 0x138

Address offset: Block B: 0x238

Address offset: Block C: 0x338

Address offset: Block D: 0x438

Address offset: Block E: 0x538

Address offset: Block F: 0x638

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DTFLKx	DTFSLKx	Res.	Res.	DTM[1:0]		SDTFx	DTFx[8:0]								
rwo	rwo			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTRLKx	DTRSLKx	Res.	DTPRSC[2:0]			SDTRx	DTRx[8:0]								
rwo	rwo		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DTFLKx**: Deadtime falling lock timer x

This write-once bit prevents the deadtime (sign and value) to be modified, if enabled.

0: Deadtime falling value and sign is writable

1: Deadtime falling value and sign is read-only

Note: This bit is not preloaded

Bit 30 **DTFSLKx**: Deadtime falling sign lock timer x

This write-once bit prevents the sign of falling deadtime to be modified, if enabled.

0: Deadtime falling sign is writable

1: Deadtime falling sign is read-only

Note: This bit is not preloaded.

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:26 **DTM[1:0]**: Deadtime mode

This bitfield defines the deadtime module behavior.

00: Deadtime complementary mode

01: Reserved

10: Reserved

11: Reserved. Writing 11 is discarded, and the DTM[1:0] value is maintained at its previous value.

Bit 25 **SDTFx**: Sign deadtime falling value timer x

This register determines whether the deadtime is positive (signals not overlapping) or negative (signals overlapping).

0: Positive deadtime on falling edge

1: Negative deadtime on falling edge

Bits 24:16 **DTFx[8:0]**: Deadtime falling value timer x

This register holds the value of the deadtime following a falling edge of reference PWM signal.

t_{DTF} = DTFx[8:0] x t_{DTG}

Bit 15 **DTRLKx**: Deadtime rising lock timer x
This write-once bit prevents the deadtime (sign and value) to be modified, if enabled.
0: Deadtime rising value and sign is writable
1: Deadtime rising value and sign is read-only
Note: This bit is not preloaded.

Bit 14 **DTRSLKx**: Deadtime rising sign lock timer x
This write-once bit prevents the sign of deadtime to be modified, if enabled.
0: Deadtime rising sign is writable
1: Deadtime rising sign is read-only
Note: This bit is not preloaded.

Bit 13 Reserved, must be kept at reset value.

Bits 12:10 **DTPRSC[2:0]**: Deadtime prescaler
This register holds the value of the deadtime clock prescaler determined by the following formula:
$t_{DTG} = (2^{(DTPRSC[2:0]-3)}) \times t_{HRTIM}$
(see *Table 342: Deadtime resolution and max absolute values*)
This bitfield is read-only as soon as any of the lock bit is enabled (DTFLKs, DTFSLKx, DTRLKx, DTRSLKx).

Bit 9 **SDTRx**: Sign deadtime rising value timer x
This register determines whether the deadtime is positive or negative (overlapping signals).
0: Positive deadtime on rising edge
1: Negative deadtime on rising edge

Bits 8:0 **DTRx[8:0]**: Deadtime rising value timer x
This register holds the value of the deadtime following a rising edge of reference PWM signal.
$t_{DTR} = DTRx[8:0] \times t_{DTG}$

33.5.27 HRTIM timer x output 1 set register (HRTIM_SETx1R) (x = A to F)

Address offset: Block A: 0x13C

Address offset: Block B: 0x23C

Address offset: Block C: 0x33C

Address offset: Block D: 0x43C

Address offset: Block E: 0x53C

Address offset: Block F: 0x63C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UPDATE**: Registers update (transfer from preload register to active register)
Register update event forces the output to its active state.

Bit 30 **EXTEVNTJ**: External event J
External event J forces the output to its active state.

Bit 29 **EXTEVNTI**: External event I
External event I forces the output to its active state.

Bit 28 **EXTEVNTH**: External event H
External event H forces the output to its active state.

Bit 27 **EXTEVNTG**: External event G
External event G forces the output to its active state.

Bit 26 **EXTEVNTF**: External event F
External event F forces the output to its active state.

Bit 25 **EXTEVNTE**: External event E
External event E forces the output to its active state.

Bit 24 **EXTEVNTD**: External event D
External event D forces the output to its active state.

Bit 23 **EXTEVNTC**: External event C
External event C forces the output to its active state.

Bit 22 **EXTEVNTB**: External event B
External event B forces the output to its active state.

Bit 21 **EXTEVNTA**: External event A
External event A forces the output to its active state.

Bit 20 **TIMEVNT9**: Timer event 9
Timers event 9 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 19 **TIMEVNT8**: Timer event 8
Timers event 8 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 18 **TIMEVNT7**: Timer event 7
Timers event 7 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 17 **TIMEVNT6**: Timer event 6
Timers event 6 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 16 **TIMEVNT5**: Timer event 5
Timers event 5 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 15 **TIMEVNT4**: Timer event 4
Timers event 4 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 14 **TIMEVNT3**: Timer event 3
Timers event 3 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 13 **TIMEVNT2**: Timer event 2
Timers event 2 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 12 **TIMEVNT1**:Timer event 1
Timers event 1 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 11 **MSTCMP4**: Master compare 4
Master timer compare 4 event forces the output to its active state.

Bit 10 **MSTCMP3**: Master compare 3
Master timer compare 3 event forces the output to its active state.

Bit 9 **MSTCMP2**: Master compare 2
Master timer compare 2 event forces the output to its active state.

Bit 8 **MSTCMP1**: Master compare 1
Master timer compare 1 event forces the output to its active state.

Bit 7 **MSTPER**: Master period
The master timer counter period event (continuous mode), or the master timer counter reset (continous or single-shot mode), forces the output to its active state.

Bit 6 **CMP4**: Timer x compare 4
Timer A compare 4 event forces the output to its active state.

Bit 5 **CMP3**: Timer x compare 3
Timer A compare 3 event forces the output to its active state.

Bit 4 **CMP2**: Timer x compare 2
Timer A compare 2 event forces the output to its active state.

Bit 3 **CMP1**: Timer x compare 1
Timer A compare 1 event forces the output to its active state.

Bit 2 **PER**: Timer x period

Timer A period event forces the output to its active state.

Note: In up/down mode (UDM bit set to 1), the counter period event is defined as per the OUTROM[1:0] bit setting.

Bit 1 **RESYNC:** Timer A resynchronization

Timer A reset event coming solely from software or SYNC input forces the output to its active state.

Note: Other timer reset are not affecting the output when RESYNC=1.

Bit 0 **SST**: Software set trigger

This bit forces the output to its active state. This bit can only be set by software and is reset by hardware.

Note: This bit is not preloaded.

33.5.28 HRTIM timer x output 1 reset register (HRTIM_RSTx1R) (x = A to F)

Address offset: Block A: 0x140

Address offset: Block B: 0x240

Address offset: Block C: 0x340

Address offset: Block D: 0x440

Address offset: Block E: 0x540

Address offset: Block F: 0x640

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UPDATE**: Registers update (transfer from preload register to active register)
Register update event forces the output to its inactive state.

Bit 30 **EXTEVNTJ**: External event J
External event J forces the output to its inactive state.

Bit 29 **EXTEVNTI**: External event I
External event I forces the output to its inactive state.

Bit 28 **EXTEVNTH**: External event H
External event H forces the output to its inactive state.

Bit 27 **EXTEVNTG**: External event G
External event G forces the output to its inactive state.

Bit 26 **EXTEVNTF**: External event F
External event F forces the output to its inactive state.

Bit 25 **EXTEVNTE**: External event E
External event E forces the output to its inactive state.

Bit 24 **EXTEVNTD**: External event D
External event D forces the output to its inactive state.

Bit 23 **EXTEVNTC**: External event C
External event C forces the output to its inactive state.

Bit 22 **EXTEVNTB**: External event B
External event B forces the output to its inactive state.

Bit 21 **EXTEVNTA**: External event A
External event A forces the output to its inactive state.

Bit 20 **TIMEVNT9**: Timer event 9
Timers event 9 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 19 **TIMEVNT8**: Timer event 8
Timers event 8 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 18 **TIMEVNT7**: Timer event 7
Timers event 7 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 17 **TIMEVNT6**: Timer event 6
Timers event 6 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 16 **TIMEVNT5**: Timer event 5
Timers event 5 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 15 **TIMEVNT4**: Timer event 4
Timers event 4 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 14 **TIMEVNT3**: Timer event 3
Timers event 3 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 13 **TIMEVNT2**: Timer event 2
Timers event 2 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 12 **TIMEVNT1**:Timer event 1
Timers event 1 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 11 **MSTCMP4**: Master compare 4
Master timer compare 4 event forces the output to its inactive state.

Bit 10 **MSTCMP3**: Master compare 3
Master timer compare 3 event forces the output to its inactive state.

Bit 9 **MSTCMP2**: Master compare 2
Master timer compare 2 event forces the output to its inactive state.

Bit 8 **MSTCMP1**: Master compare 1
Master timer compare 1 event forces the output to its inactive state.

Bit 7 **MSTPER**: Master period
The master timer counter period event (continuous mode), or the master timer counter reset (continous or single-shot mode), forces the output to its inactive state.

Bit 6 **CMP4**: Timer x compare 4
Timer A compare 4 event forces the output to its inactive state.

Bit 5 **CMP3**: Timer x compare 3
Timer A compare 3 event forces the output to its inactive state.

Bit 4 **CMP2**: Timer x compare 2
Timer A compare 2 event forces the output to its inactive state.

Bit 3 **CMP1**: Timer x compare 1
Timer A compare 1 event forces the output to its inactive state.

Bit 2 **PER**: Timer x period

Timer A period event forces the output to its inactive state.

Note: In up/down mode (UDM bit set to 1), the counter period event is defined as per the OUTROM[1:0] bit setting.

Bit 1 **RESYNC:** Timer A resynchronization

Timer A reset event coming solely from software or SYNC input forces the output to its inactive state.

Note: Other timer reset are not affecting the output when RESYNC=1.

Bit 0 **SRT**: Software reset trigger

This bit forces the output to its inactive state. This bit can only be set by software and is reset by hardware.

Note: This bit is not preloaded.

33.5.29 HRTIM timer x output 2 set register (HRTIM_SETx2R) (x = A to F)

Address offset: Block A: 0x144

Address offset: Block B: 0x244

Address offset: Block C: 0x344

Address offset: Block D: 0x444

Address offset: Block E: 0x544

Address offset: Block F: 0x644

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UPDATE**: Registers update (transfer from preload register to active register)

Register update event forces the output to its active state.

Bit 30 **EXTEVNTJ**: External event J

External event J forces the output to its active state.

Bit 29 **EXTEVNTI**: External event I

External event I forces the output to its active state.

Bit 28 **EXTEVNTH**: External event H

External event H forces the output to its active state.

Bit 27 **EXTEVNTG**: External event G

External event G forces the output to its active state.

Bit 26 **EXTEVNTF**: External event F

External event F forces the output to its active state.

Bit 25 **EXTEVNTE**: External event E
External event E forces the output to its active state.

Bit 24 **EXTEVNTD**: External event D
External event D forces the output to its active state.

Bit 23 **EXTEVNTC**: External event C
External event C forces the output to its active state.

Bit 22 **EXTEVNTB**: External event B
External event B forces the output to its active state.

Bit 21 **EXTEVNTA**: External event A
External event A forces the output to its active state.

Bit 20 **TIMEVNT9**: Timer event 9
Timers event 9 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 19 **TIMEVNT8**: Timer event 8
Timers event 8 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 18 **TIMEVNT7**: Timer event 7
Timers event 7 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 17 **TIMEVNT6**: Timer event 6
Timers event 6 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 16 **TIMEVNT5**: Timer event 5
Timers event 5 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 15 **TIMEVNT4**: Timer event 4
Timers event 4 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 14 **TIMEVNT3**: Timer event 3
Timers event 3 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 13 **TIMEVNT2**: Timer event 2
Timers event 2 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 12 **TIMEVNT1**:Timer event 1
Timers event 1 forces the output to its active state (refer to *Table 339* for timer events assignments).

Bit 11 **MSTCMP4**: Master compare 4
Master timer compare 4 event forces the output to its active state.

Bit 10 **MSTCMP3**: Master compare 3
Master timer compare 3 event forces the output to its active state.

Bit 9 **MSTCMP2**: Master compare 2
Master timer compare 2 event forces the output to its active state.

Bit 8 **MSTCMP1**: Master compare 1
Master timer compare 1 event forces the output to its active state.

Bit 7 **MSTPER**: Master period
The master timer counter period event (continuous mode), or the master timer counter reset (continous or single-shot mode), forces the output to its active state.

Bit 6 **CMP4**: Timer x compare 4
Timer A compare 4 event forces the output to its active state.

Bit 5 **CMP3**: Timer x compare 3

Timer A compare 3 event forces the output to its active state.

Bit 4 **CMP2**: Timer x compare 2

Timer A compare 2 event forces the output to its active state.

Bit 3 **CMP1**: Timer x compare 1

Timer A compare 1 event forces the output to its active state.

Bit 2 **PER**: Timer x period

Timer A period event forces the output to its active state.

Note: In up/down mode (UDM bit set to 1), the counter period event is defined as per the OUTROM[1:0] bit setting.

Bit 1 **RESYNC:** Timer A resynchronization

Timer A reset event coming solely from software or SYNC input forces the output to its active state.

Note: Other timer reset are not affecting the output when RESYNC=1.

Bit 0 **SST**: Software set trigger

This bit forces the output to its active state. This bit can only be set by software and is reset by hardware.

Note: This bit is not preloaded.

33.5.30 HRTIM timer x output 2 reset register (HRTIM_RSTx2R) (x = A to F)

Address offset: Block A: 0x148

Address offset: Block B: 0x248

Address offset: Block C: 0x348

Address offset: Block D: 0x448

Address offset: Block E: 0x548

Address offset: Block F: 0x648

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UPDATE**: Registers update (transfer from preload register to active register)
Register update event forces the output to its inactive state.

Bit 30 **EXTEVNTJ**: External event J
External event J forces the output to its inactive state.

Bit 29 **EXTEVNTI**: External event I
External event I forces the output to its inactive state.

Bit 28 **EXTEVNTH**: External event H
External event H forces the output to its inactive state.

Bit 27 **EXTEVNTG**: External event G
External event G forces the output to its inactive state.

Bit 26 **EXTEVNTF**: External event F
External event F forces the output to its inactive state.

Bit 25 **EXTEVNTE**: External event E
External event E forces the output to its inactive state.

Bit 24 **EXTEVNTD**: External event D
External event D forces the output to its inactive state.

Bit 23 **EXTEVNTC**: External event C
External event C forces the output to its inactive state.

Bit 22 **EXTEVNTB**: External event B
External event B forces the output to its inactive state.

Bit 21 **EXTEVNTA**: External event A
External event A forces the output to its inactive state.

Bit 20 **TIMEVNT9**: Timer event 9
Timers event 9 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 19 **TIMEVNT8**: Timer event 8
Timers event 8 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 18 **TIMEVNT7**: Timer event 7
Timers event 7 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 17 **TIMEVNT6**: Timer event 6
Timers event 6 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 16 **TIMEVNT5**: Timer event 5
Timers event 5 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 15 **TIMEVNT4**: Timer event 4
Timers event 4 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 14 **TIMEVNT3**: Timer event 3
Timers event 3 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 13 **TIMEVNT2**: Timer event 2
Timers event 2 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 12 **TIMEVNT1**:Timer event 1
Timers event 1 forces the output to its inactive state (refer to *Table 339* for timer events assignments).

Bit 11 **MSTCMP4**: Master compare 4
Master timer compare 4 event forces the output to its inactive state.

Bit 10 **MSTCMP3**: Master compare 3
Master timer compare 3 event forces the output to its inactive state.

Bit 9 **MSTCMP2**: Master compare 2
Master timer compare 2 event forces the output to its inactive state.

Bit 8 **MSTCMP1**: Master compare 1
Master timer compare 1 event forces the output to its inactive state.

Bit 7 **MSTPER**: Master period
The master timer counter period event (continuous mode), or the master timer counter reset (continous or single-shot mode), forces the output to its inactive state.

Bit 6 **CMP4**: Timer x compare 4
Timer A compare 4 event forces the output to its inactive state.

Bit 5 **CMP3**: Timer x compare 3
Timer A compare 3 event forces the output to its inactive state.

Bit 4 **CMP2**: Timer x compare 2
Timer A compare 2 event forces the output to its inactive state.

Bit 3 **CMP1**: Timer x compare 1
Timer A compare 1 event forces the output to its inactive state.

Bit 2 **PER**: Timer x period

Timer A period event forces the output to its inactive state.

Note: In up/down mode (UDM bit set to 1), the counter period event is defined as per the OUTROM[1:0] bit setting.

Bit 1 **RESYNC:** Timer A resynchronization

Timer A reset event coming solely from software or SYNC input forces the output to its inactive state.

Note: Other timer reset are not affecting the output when RESYNC=1.

Bit 0 **SRT**: Software reset trigger

This bit forces the output to its inactive state. This bit can only be set by software and is reset by hardware.

Note: This bit is not preloaded.

33.5.31 HRTIM timer x external event filtering register 1 (HRTIM_EEFxR1) (x = A to F)

Address offset: Block A: 0x14C

Address offset: Block B: 0x24C

Address offset: Block C: 0x34C

Address offset: Block D: 0x44C

Address offset: Block E: 0x54C

Address offset: Block F: 0x64C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EEEFLTR[4:0]					EEELTCH	EEDFLTR[4:0]					EEDLTCH	EECFLTR[4:3]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EECFLTR[2:0]			EECLTCH	EEBFLTR[4:0]					EEBLTCH	EEAFLTR[4:0]					EEALTCH
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:25 **EEEFLTR[4:0]**: External event E filter
Refer to EEAFLTR[3:0] description.

Bit 24 **EEELTCH**: External event E latch
Refer to EEALTCH description

Bits 23:19 **EEDFLTR[4:0]**: External event D filter
Refer to EEAFLTR[3:0] description.

Bit 18 **EEDLTCH**: External event D latch
Refer to EEALTCH description

Bits 17:13 **EECFLTR[4:0]**: External event C filter
Refer to EEAFLTR[3:0] description.

Bit 12 **EECLTCH**: External event C latch
Refer to EEALTCH description

Bits 11:7 **EEBFLTR[4:0]**: External event B filter
Refer to EEAFLTR[3:0] description.

Bit 6 **EEBLTCH**: External event B latch
Refer to EEALTCH description.

Bits 5:1 **EEAFLTR[4:0]**: External event A filter

00000: No filtering

00001: Blanking from counter reset/roll-over to compare 1

00010: Blanking from counter reset/roll-over to compare 2 in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): blanking from compare 1 to compare 2, only during the up-counting phase.

00011: Blanking from counter reset/roll-over to compare 3

00100: Blanking from counter reset/roll-over to compare 4 in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): blanking from compare 3 to compare 4, only during the up-counting phase.

0XXXX: Blanking from another timing unit: TIMFLTRy source (see *Table 348* for details)
XXXX = 0101 → y = 1
...
XXXX = 1100 → y = 8

01101: Windowing from counter reset/roll-over to compare 2 in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): windowing from compare 2 to compare 3, only during the up-counting phase.

01110: Windowing from counter reset/roll-over to compare 3 in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): windowing from compare 2 to compare 3, only during the down-counting phase.

01111: Windowing from another timing unit: TIMWIN source (see *Table 349* for details) in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): windowing from compare 2 during the up-counting phase to compare 3 during the down-counting phase.

10000: Blanking from compare 1 to counter reset/roll-over (UDM bit reset).In up-down counting mode (UDM bit set): blanking from compare 1 during down-counting to compare 2 during the following up-counting phase.

10001: Blanking from compare 2 to counter reset/roll-over (UDM bit reset).In up-down counting mode (UDM bit set): blanking from CMP1 (down-counting) to CMP2 (down-counting).

10010: Blanking from compare 1 to compare 2 (UDM bit reset). In up-down counting mode (UDM bit set): blanking from CMP1 (up-counting) to CMP2 (down-counting.

10011: Blanking from compare 3 to compare 4 (UDM bit reset). In up-down counting mode (UDM bit set): blanking from CMP3 (up-counting) to CMP4 (down-counting).

10100: Windowing from compare 1 to counter reset/roll-over. In up-down counting mode (UDM bit set), the reset/roll-over event is conditioned to the ROM[1:0] bitfield setting

10101: Windowing from compare 2 to counter reset/roll-over. In up-down counting mode (UDM bit set), the reset/roll-over event is conditioned to the ROM[1:0] bitfield setting

10110: Windowing from compare 1 to compare 2 (UDM bit reset). In up-down counting mode (UDM bit set): windowing from CMP1 (up-counting) to CMP2 (down-counting).

10111: Windowing from compare 3 to compare 4 (UDM bit reset). In up-down counting mode (UDM bit set): windowing from compare 3 during down-counting to compare 4 during the following up-counting phase

11000: TIMWIN2 source (see *Table 349* for details). In up-down counting mode (UDM bit set), the reset/roll-over event is conditioned to the ROM[1:0] bitfield setting

11001: TIMWIN3 source (see *Table 349* for details)
In up-down counting mode (UDM bit set): windowing from compare 1 during up-counting to compare 2 during the down-counting.

11010: Blanking from Period Event to Compare 1 In up-down counting mode (UDM bit set), the reset/roll-over event is conditioned to the ROM[1:0] bitfield setting

11011: Blanking from Period Event to Compare 2 In up-down counting mode (UDM bit set), the reset/roll-over event is conditioned to the ROM[1:0] bitfield setting

Note: Whenever a compare register is used for filtering, the value must be strictly above 0.
This bitfield must not be modified once the counter is enabled (TxCEN bit set)

Bit 0 **EEALTCH**: External event A latch

0: Event A is ignored if it happens during a blank, or passed through during a window.

1: Event A is latched and delayed till the end of the blanking or windowing period.

Note: A timeout event is generated in window mode (EEAFLTR[3:0]=1101, 1110, 1111) if EEALTCH = 0, except if the external event is programmed in fast mode (EExFAST = 1).

This bitfield must not be modified once the counter is enabled (TxCEN bit set).

33.5.32 HRTIM timer x external event filtering register 2 (HRTIM_EEFxR2) (x = A to F)

Address offset: Block A: 0x150

Address offset: Block B: 0x250

Address offset: Block C: 0x350

Address offset: Block D: 0x450

Address offset: Block E: 0x550

Address offset: Block F: 0x650

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EEJFLTR[4:0]					EEJLTCH	EEIFLTR[4:0]					EEILTCH	EEHFLTR[4:3]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EEHFLTR[2:0]			EEHLTCH	EEGFLTR[4:0]					EEGLTCH	EEFFLTR[4:0]					EEFLTCH
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:25 **EEJFLTR[4:0]**: External event J filter
Refer to EEAFLTR[3:0] description.

Bit 24 **EEJLTCH**: External event J latch
Refer to EEALTCH description.

Bits 23:19 **EEIFLTR[4:0]**: External event I filter
Refer to EEAFLTR[3:0] description.

Bit 18 **EEILTCH**: External event I latch
Refer to EEALTCH description.

Bits 17:13 **EEHFLTR[4:0]**: External event H filter
Refer to EEAFLTR[3:0] description.

Bit 12 **EEHLTCH**: External event H latch
Refer to EEALTCH description.

Bits 11:7 **EEGFLTR[4:0]**: External event G filter
Refer to EEAFLTR[3:0] description.

Bit 6 **EEGLTCH**: External event G latch
Refer to EEALTCH description.

Bits 5:1 **EEFFLTR[4:0]**: External event F filter
Refer to EEAFLTR[3:0] description.

Bit 0 **EEFLTCH**: External event F latch
Refer to EEALTCH description.

33.5.33 HRTIM timer A reset register (HRTIM_RSTAR)

Address offset: 0x154

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCMP2**: Timer F compare 2
The timer A counter is reset upon timer F Compare 2 event.

Bit 30 **TIMECMP4**: Timer E compare 4
The timer A counter is reset upon timer E compare 4 event.

Bit 29 **TIMECMP2**: Timer E compare 2
The timer A counter is reset upon timer E compare 2 event.

Bit 28 **TIMECMP1**: Timer E compare 1
The timer A counter is reset upon timer E compare 1 event.

Bit 27 **TIMDCMP4**: Timer D compare 4
The timer A counter is reset upon timer D compare 4 event.

Bit 26 **TIMDCMP2**: Timer D compare 2
The timer A counter is reset upon timer D compare 2 event.

Bit 25 **TIMDCMP1**: Timer D compare 1
The timer A counter is reset upon timer D compare 1 event.

Bit 24 **TIMCCMP4**: Timer C compare 4
The timer A counter is reset upon timer C compare 4 event.

Bit 23 **TIMCCMP2**: Timer C compare 2
The timer A counter is reset upon timer C compare 2 event.

Bit 22 **TIMCCMP1**: Timer C compare 1
The timer A counter is reset upon timer C compare 1 event.

Bit 21 **TIMBCMP4**: Timer B compare 4
The timer A counter is reset upon timer B compare 4 event.

Bit 20 **TIMBCMP2**: Timer B compare 2
The timer A counter is reset upon timer B compare 2 event.

Bit 19 **TIMBCMP1**: Timer B compare 1
The timer A counter is reset upon timer B compare 1 event.

Bit 18 **EXTEVNTJ**: External event J
The timer A counter is reset upon external event J.

Bit 17 **EXTEVNTI**: External event I
The timer A counter is reset upon external event I.

Bit 16 **EXTEVNTH**: External event H
The timer A counter is reset upon external event H.

Bit 15 **EXTEVNTG**: External event G
The timer A counter is reset upon external event G.

Bit 14 **EXTEVNTF**: External event F
The timer A counter is reset upon external event F.

Bit 13 **EXTEVNTE**: External event E
The timer A counter is reset upon external event E.

Bit 12 **EXTEVNTD**: External event D
The timer A counter is reset upon external event D.

Bit 11 **EXTEVNTC**: External event C
The timer A counter is reset upon external event C.

Bit 10 **EXTEVNTB**: External event B
The timer A counter is reset upon external event B.

Bit 9 **EXTEVNTA**: External event A
The timer A counter is reset upon external event A.

Bit 8 **MSTCMP4**: Master compare 4
The timer A counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer A counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer A counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer A counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER**: Master timer period
The timer A counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer A compare 4 reset
The timer A counter is reset upon timer A compare 4 event.

Bit 2 **CMP2**: Timer A compare 2 reset
The timer A counter is reset upon timer A compare 2 event.

Bit 1 **UPDT**: Timer A update reset
The timer A counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer A counter is reset upon timer F compare 1 event.

33.5.34 HRTIM timer B reset register (HRTIM_RSTBR)

Address offset: 0x254

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCMP2**: Timer F compare 2
The timer B counter is reset upon timer F Compare 2 event.

Bit 30 **TIMECMP4**: Timer E compare 4
The timer B counter is reset upon timer E compare 4 event.

Bit 29 **TIMECMP2**: Timer E compare 2
The timer B counter is reset upon timer E compare 2 event.

Bit 28 **TIMECMP1**: Timer E compare 1
The timer B counter is reset upon timer E compare 1 event.

Bit 27 **TIMDCMP4**: Timer D compare 4
The timer B counter is reset upon timer D compare 4 event.

Bit 26 **TIMDCMP2**: Timer D compare 2
The timer B counter is reset upon timer D compare 2 event.

Bit 25 **TIMDCMP1**: Timer D compare 1
The timer B counter is reset upon timer D compare 1 event.

Bit 24 **TIMCCMP4**: Timer C compare 4
The timer B counter is reset upon timer C compare 4 event.

Bit 23 **TIMCCMP2**: Timer C compare 2
The timer B counter is reset upon timer C compare 2 event.

Bit 22 **TIMCCMP1**: Timer C compare 1
The timer B counter is reset upon timer C compare 1 event.

Bit 21 **TIMACMP4**: Timer A compare 4
The timer B counter is reset upon timer A compare 4 event.

Bit 20 **TIMACMP2**: Timer A compare 2
The timer B counter is reset upon timer A compare 2 event.

Bit 19 **TIMACMP1**: Timer A compare 1
The timer B counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNTJ**: External event J
The timer B counter is reset upon external event J.

Bit 17 **EXTEVNTI**: External event I
The timer B counter is reset upon external event I.

Bit 16 **EXTEVNTH**: External event H
The timer B counter is reset upon external event H.

Bit 15 **EXTEVNTG**: External event G
The timer B counter is reset upon external event G.

Bit 14 **EXTEVNTF**: External event F
The timer B counter is reset upon external event F.

Bit 13 **EXTEVNTE**: External event E
The timer B counter is reset upon external event E.

Bit 12 **EXTEVNTD**: External event D
The timer B counter is reset upon external event D.

Bit 11 **EXTEVNTC**: External event C
The timer B counter is reset upon external event C.

Bit 10 **EXTEVNTB**: External event B
The timer B counter is reset upon external event B.

Bit 9 **EXTEVNTA**: External event A
The timer B counter is reset upon external event A.

Bit 8 **MSTCMP4**: Master compare 4
The timer B counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer B counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer B counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer B counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER**: Master timer period
The timer B counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer B compare 4 reset
The timer B counter is reset upon timer B compare 4 event.

Bit 2 **CMP2**: Timer B compare 2 reset
The timer B counter is reset upon timer B compare 2 event.

Bit 1 **UPDT**: Timer B update reset
The timer B counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer B counter is reset upon timer F compare 1 event.

33.5.35 HRTIM timer C reset register (HRTIM_RSTCR)

Address offset: 0x354

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCMP2**: Timer F compare 2
The timer C counter is reset upon timer F Compare 2 event.

Bit 30 **TIMECMP4**: Timer E compare 4
The timer C counter is reset upon timer E compare 4 event.

Bit 29 **TIMECMP2**: Timer E compare 2
The timer C counter is reset upon timer E compare 2 event.

Bit 28 **TIMECMP1**: Timer E compare 1
The timer C counter is reset upon timer E compare 1 event.

Bit 27 **TIMDCMP4**: Timer D compare 4
The timer C counter is reset upon timer D compare 4 event.

Bit 26 **TIMDCMP2**: Timer D compare 2
The timer C counter is reset upon timer D compare 2 event.

Bit 25 **TIMDCMP1**: Timer D compare 1
The timer C counter is reset upon timer D compare 1 event.

Bit 24 **TIMBCMP4**: Timer B compare 4
The timer C counter is reset upon timer B compare 4 event.

Bit 23 **TIMBCMP2**: Timer B compare 2
The timer C counter is reset upon timer B compare 2 event.

Bit 22 **TIMBCMP1**: Timer B compare 1
The timer C counter is reset upon timer B compare 1 event.

Bit 21 **TIMACMP4**: Timer A compare 4
The timer C counter is reset upon timer A compare 4 event.

Bit 20 **TIMACMP2**: Timer A compare 2
The timer C counter is reset upon timer A compare 2 event.

Bit 19 **TIMACMP1**: Timer A compare 1
The timer C counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNTJ**: External event J
The timer C counter is reset upon external event J.

Bit 17 **EXTEVNTI**: External event I
The timer C counter is reset upon external event I.

Bit 16 **EXTEVNTH**: External event H
The timer C counter is reset upon external event H.

Bit 15 **EXTEVNTG**: External event G
The timer C counter is reset upon external event G.

Bit 14 **EXTEVNTF**: External event F
The timer C counter is reset upon external event F.

Bit 13 **EXTEVNTE**: External event E
The timer C counter is reset upon external event E.

Bit 12 **EXTEVNTD**: External event D
The timer Ccounter is reset upon external event D.

Bit 11 **EXTEVNTC**: External event C
The timer C counter is reset upon external event C.

Bit 10 **EXTEVNTB**: External event B
The timer C counter is reset upon external event B.

Bit 9 **EXTEVNTA**: External event A
The timer C counter is reset upon external event A.

Bit 8 **MSTCMP4**: Master compare 4
The timer C counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer C counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer C counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer C counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER**: Master timer period
The timer C counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer C compare 4 reset
The timer C counter is reset upon timer C compare 4 event.

Bit 2 **CMP2**: Timer C compare 2 reset
The timer C counter is reset upon timer C compare 2 event.

Bit 1 **UPDT**: Timer C update reset
The timer C counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer C counter is reset upon timer F compare 1 event.

33.5.36 HRTIM timer D reset register (HRTIM_RSTDR)

Address offset: 0x454

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCMP2**: Timer F compare 2
The timer D counter is reset upon timer F Compare 2 event.

Bit 30 **TIMECMP4**: Timer E compare 4
The timer D counter is reset upon timer E compare 4 event.

Bit 29 **TIMECMP2**: Timer E compare 2
The timer D counter is reset upon timer E compare 2 event.

Bit 28 **TIMECMP1**: Timer E compare 1
The timer D counter is reset upon timer E compare 1 event.

Bit 27 **TIMCCMP4**: Timer C compare 4
The timer D counter is reset upon timer C compare 4 event.

Bit 26 **TIMCCMP2**: Timer C compare 2
The timer D counter is reset upon timer C compare 2 event.

Bit 25 **TIMCCMP1**: Timer C compare 1
The timer D counter is reset upon timer C compare 1 event.

Bit 24 **TIMBCMP4**: Timer B compare 4
The timer D counter is reset upon timer B compare 4 event.

Bit 23 **TIMBCMP2**: Timer B compare 2
The timer D counter is reset upon timer B compare 2 event.

Bit 22 **TIMBCMP1**: Timer B compare 1
The timer D counter is reset upon timer B compare 1 event.

Bit 21 **TIMACMP4**: Timer A compare 4
The timer D counter is reset upon timer A compare 4 event.

Bit 20 **TIMACMP2**: Timer A compare 2
The timer D counter is reset upon timer A compare 2 event.

Bit 19 **TIMACMP1**: Timer A compare 1
The timer D counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNTJ**: External event J
The timer A counter is reset upon external event J.

Bit 17 **EXTEVNTI**: External event I
The timer A counter is reset upon external event I.

Bit 16 **EXTEVNTH**: External event H
The timer A counter is reset upon external event H.

Bit 15 **EXTEVNTG**: External event G
The timer A counter is reset upon external event G.

Bit 14 **EXTEVNTF**: External event F
The timer A counter is reset upon external event F.

Bit 13 **EXTEVNTE**: External event E
The timer A counter is reset upon external event E.

Bit 12 **EXTEVNTD**: External event D
The timer A counter is reset upon external event D.

Bit 11 **EXTEVNTC**: External event C
The timer A counter is reset upon external event C.

Bit 10 **EXTEVNTB**: External event B
The timer A counter is reset upon external event B.

Bit 9 **EXTEVNTA**: External event A
The timer A counter is reset upon external event A.

Bit 8 **MSTCMP4**: Master compare 4
The timer D counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer D counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer D counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer D counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER**: Master timer period
The timer D counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer D compare 4 reset
The timer D counter is reset upon timer D compare 4 event.

Bit 2 **CMP2**: Timer D compare 2 reset
The timer D counter is reset upon timer D compare 2 event.

Bit 1 **UPDT**: Timer D update reset
The timer D counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer D counter is reset upon timer F compare 1 event.

33.5.37 HRTIM timer E reset register (HRTIM_RSTER)

Address offset: 0x554

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMFCMP2	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCMP2**: Timer F compare 2
The timer E counter is reset upon timer F Compare 2 event.

Bit 30 **TIMDCMP4**: Timer D compare 4
The timer E counter is reset upon timer D compare 4 event.

Bit 29 **TIMDCMP2**: Timer D compare 2
The timer E counter is reset upon timer D compare 2 event.

Bit 28 **TIMDCMP1**: Timer D compare 1
The timer E counter is reset upon timer D compare 1 event.

Bit 27 **TIMCCMP4**: Timer C compare 4
The timer E counter is reset upon timer C compare 4 event.

Bit 26 **TIMCCMP2**: Timer C compare 2
The timer E counter is reset upon timer C compare 2 event.

Bit 25 **TIMCCMP1**: Timer C compare 1
The timer E counter is reset upon timer C compare 1 event.

Bit 24 **TIMBCMP4**: Timer B compare 4
The timer E counter is reset upon timer B compare 4 event.

Bit 23 **TIMBCMP2**: Timer B compare 2
The timer E counter is reset upon timer B compare 2 event.

Bit 22 **TIMBCMP1**: Timer B compare 1
The timer E counter is reset upon timer B compare 1 event.

Bit 21 **TIMACMP4**: Timer A compare 4
The timer E counter is reset upon timer A compare 4 event.

Bit 20 **TIMACMP2**: Timer A compare 2
The timer E counter is reset upon timer A compare 2 event.

Bit 19 **TIMACMP1**: Timer A compare 1
The timer E counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNTJ**: External event J
The timer E counter is reset upon external event J.

Bit 17 **EXTEVNTI**: External event I
The timer E counter is reset upon external event I.

Bit 16 **EXTEVNTH**: External event H
The timer E counter is reset upon external event H.

Bit 15 **EXTEVNTG**: External event G
The timer E counter is reset upon external event G.

Bit 14 **EXTEVNTF**: External event F
The timer E counter is reset upon external event F.

Bit 13 **EXTEVNTE**: External event E
The timer E counter is reset upon external event E.

Bit 12 **EXTEVNTD**: External event D
The timer E counter is reset upon external event D.

Bit 11 **EXTEVNTC**: External event C
The timer E counter is reset upon external event C.

Bit 10 **EXTEVNTB**: External event B
The timer E counter is reset upon external event B.

Bit 9 **EXTEVNTA**: External event A
The timer E counter is reset upon external event A.

Bit 8 **MSTCMP4**: Master compare 4
The timer E counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer E counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer E counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer E counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER**: Master timer period
The timer E counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer E compare 4 reset
The timer E counter is reset upon timer E compare 4 event.

Bit 2 **CMP2**: Timer E compare 2 reset
The timer E counter is reset upon timer E compare 2 event.

Bit 1 **UPDT**: Timer E update reset
The timer E counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer E counter is reset upon timer F compare 1 event.

33.5.38 HRTIM timer F reset register (HRTIM_RSTFR)

Address offset: 0x654

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMECMP2	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMECMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMECMP2**: Timer E compare 2
The timer F counter is reset upon timer E Compare 2 event.

Bit 30 **TIMDCMP4**: Timer D compare 4
The timer F counter is reset upon timer D compare 4 event.

Bit 29 **TIMDCMP2**: Timer D compare 2
The timer F counter is reset upon timer D compare 2 event.

Bit 28 **TIMDCMP1**: Timer D compare 1
The timer F counter is reset upon timer D compare 1 event.

Bit 27 **TIMCCMP4**: Timer C compare 4
The timer F counter is reset upon timer C compare 4 event.

Bit 26 **TIMCCMP2**: Timer C compare 2
The timer F counter is reset upon timer C compare 2 event.

Bit 25 **TIMCCMP1**: Timer C compare 1
The timer F counter is reset upon timer C compare 1 event.

Bit 24 **TIMBCMP4**: Timer B compare 4
The timer F counter is reset upon timer B compare 4 event.

Bit 23 **TIMBCMP2**: Timer B compare 2
The timer F counter is reset upon timer B compare 2 event.

Bit 22 **TIMBCMP1**: Timer B compare 1
The timer F counter is reset upon timer B compare 1 event.

Bit 21 **TIMACMP4**: Timer A compare 4
The timer F counter is reset upon timer A compare 4 event.

Bit 20 **TIMACMP2**: Timer A compare 2
The timer F counter is reset upon timer A compare 2 event.

Bit 19 **TIMACMP1**: Timer A compare 1
The timer F counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNTJ**: External event J
The timer F counter is reset upon external event J.

Bit 17 **EXTEVNTI**: External event I
The timer F counter is reset upon external event I.

Bit 16 **EXTEVNTH**: External event H
The timer F counter is reset upon external event H.

Bit 15 **EXTEVNTG**: External event G
The timer F counter is reset upon external event G.

Bit 14 **EXTEVNTF**: External event F
The timer F counter is reset upon external event F.

Bit 13 **EXTEVNTE**: External event E
The timer F counter is reset upon external event E.

Bit 12 **EXTEVNTD**: External event D
The timer F counter is reset upon external event D.

Bit 11 **EXTEVNTC**: External event C
The timer F counter is reset upon external event C.

Bit 10 **EXTEVNTB**: External event B
The timer F counter is reset upon external event B.

Bit 9 **EXTEVNTA**: External event A
The timer F counter is reset upon external event A.

Bit 8 **MSTCMP4**: Master compare 4
The timer F counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer F counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer F counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer F counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER**: Master timer period
The timer F counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer F compare 4 reset
The timer F counter is reset upon timer F compare 4 event.

Bit 2 **CMP2**: Timer F compare 2 reset
The timer F counter is reset upon timer F compare 2 event.

Bit 1 **UPDT**: Timer F update reset
The timer F counter is reset upon update event.

Bit 0 **TIMECMP1**: Timer E compare 1
The timer F counter is reset upon timer E compare 1 event.

33.5.39 HRTIM timer x chopper register (HRTIM_CHPxR) (x = A to F)

Address offset: Block A: 0x158

Address offset: Block B: 0x258

Address offset: Block C: 0x358

Address offset: Block D: 0x458

Address offset: Block E: 0x558

Address offset: Block F: 0x658

Reset value: 0x0000 0000

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

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:7 **STRTPW[3:0]**: Timer x start pulsewidth

This register defines the initial pulsewidth following a rising edge on output signal.
This bitfield cannot be modified when one of the CHPx bits is set.
t_{1STPW} = (STRPW[3:0]+1) x 16 x t_{HRTIM}.
0000: 31 ns (1/32.5 MHz)
...
1111: 492 ns (16/32.5 MHz)

Bits 6:4 **CARDTY[2:0]**: Timer x chopper duty cycle value

This register defines the duty cycle of the carrier signal. This bitfield cannot be modified when one of the CHPx bits is set.
000: 0/8 (i.e. only 1st pulse is present)
...
111: 7/8

Bits 3:0 **CARFRQ[3:0]**: Timer x carrier frequency value

This register defines the carrier frequency F_{CHPFRQ} = f_{HRTIM} / (16 x (CARFRQ[3:0]+1)).
This bitfield cannot be modified when one of the CHPx bits is set.
0000: 32.5 MHz (f_{HRTIM}/ 16)
...
1111: 2.03 MHz (f_{HRTIM} / 256)

33.5.40 HRTIM timer A capture 1 control register (HRTIM_CPT1ACR)

Address offset: 0x15C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TFCMP2	TFCMP1	TF1RST	TF1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TFCMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 14 **TFCMP1**: Timer F compare 1
Refer to TACMP1 description.

Bit 13 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 12 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 1

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 1

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 1 by software. This bit is set only, reset by hardware.

33.5.41 HRTIM timer B capture 1 control register (HRTIM_CPT1BCR)

Address offset: 0x25C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TFCMP2	TFCMP1	TF1RST	TF1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TFCMP2**: Timer F compare 2
Refer to TACMP2 description.

Bit 18 **TFCMP1**: Timer F compare 1
Refer to TACMP1 description.

Bit 17 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 16 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 1

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture1

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 1 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 1 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 1

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 1

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 1 by software. This bit is set only, reset by hardware.

33.5.42 HRTIM timer C capture 1 control register (HRTIM_CPT1CCR)

Address offset: 0x35C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TFCMP2**: Timer F compare 2
Refer to TACMP2 description.

Bit 22 **TFCMP1**:Timer F compare 1
Refer to TACMP1 description.

Bit 21 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 20 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 1

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 1

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 1 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 1 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 1

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 1

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 1 by software. This bit is set only, reset by hardware.

33.5.43 HRTIM timer D capture 1 control register (HRTIM_CPT1DCR)

Address offset: 0x45C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TFCMP2**: Timer F compare 2
Refer to TACMP2 description.

Bit 26 **TFCMP1**: Timer F compare 1
Refer to TACMP1 description.

Bit 25 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 24 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 1

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 1

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 1 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 1 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 1

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 1

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 1 by software. This bit is set only, reset by hardware.

33.5.44 HRTIM timer E capture 1 control register (HRTIM_CPT1ECR)

Address offset: 0x55C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TFCMP2	TFCMP1	TF1RST	TF1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TFCMP2**: Timer F compare 2
Refer to TACMP2 description.

Bit 30 **TFCMP1**: Timer F compare 1
Refer to TACMP1 description.

Bit 29 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 28 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 1

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 1

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 1 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 1 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 1

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 1

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 1 by software. This bit is set only, reset by hardware.

33.5.45 HRTIM timer F capture 1 control register (HRTIM_CPT1FCR)

Address offset: 0x65C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 1

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 1

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 1 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 1 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 1

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 1

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 1 by software. This bit is set only, reset by hardware.

33.5.46 HRTIM timer A capture 2 control register (HRTIM_CPT2ACR)

Address offset: 0x160

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TFCMP2	TFCMP1	TF1RST	TF1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TFCMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 14 **TFCMP1**: Timer F compare 1
Refer to TACMP1 description.

Bit 13 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 12 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 2

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 2

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 2 by software. This bit is set only, reset by hardware.

33.5.47 HRTIM timer B capture 2 control register (HRTIM_CPT2BCR)

Address offset: 0x260

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TFCMP2	TFCMP1	TF1RST	TF1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TFCMP2**: Timer F compare 2
Refer to TACMP2 description.

Bit 18 **TFCMP1**: Timer F compare 1
Refer to TACMP1 description.

Bit 17 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 16 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 2

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 2

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 2 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 2 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 2

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 2

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 2 by software. This bit is set only, reset by hardware.

33.5.48 HRTIM timer C capture 2 control register (HRTIM_CPT2CCR)

Address offset: 0x360

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TFCMP2**: Timer F compare 2
Refer to TACMP2 description.

Bit 22 **TFCMP1**:Timer F compare 1
Refer to TACMP1 description.

Bit 21 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 20 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 2

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 2

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 2 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 2 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 2

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 2

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 2 by software. This bit is set only, reset by hardware.

33.5.49 HRTIM timer D capture 2 control register (HRTIM_CPT2DCR)

Address offset: 0x460

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TFCMP2**: Timer F compare 2
Refer to TACMP2 description.

Bit 26 **TFCMP1**: Timer F compare 1
Refer to TACMP1 description.

Bit 25 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 24 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 2

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 2

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 2 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 2 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 2

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 2

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 2 by software. This bit is set only, reset by hardware.

33.5.50 HRTIM timer E capture 2 control register (HRTIM_CPT2ECR)

Address offset: 0x560

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TFCMP2	TFCMP1	TF1RST	TF1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TFCMP2**: Timer F compare 2
Refer to TACMP2 description.

Bit 30 **TFCMP1**: Timer F compare 1
Refer to TACMP1 description.

Bit 29 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 28 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 2

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 2

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 2 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 2 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 2

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 2

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 2 by software. This bit is set only, reset by hardware.

33.5.51 HRTIM timer F capture 2 control register (HRTIM_CPT2FCR)

Address offset: 0x660

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 2

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 2

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 2 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 2 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 2

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 2

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 2 by software. This bit is set only, reset by hardware.

33.5.52 HRTIM timer x output register (HRTIM_OUTxR) (x = A to F)

Address offset: Block A: 0x164

Address offset: Block B: 0x264

Address offset: Block C: 0x364

Address offset: Block D: 0x464

Address offset: Block E: 0x564

Address offset: Block F: 0x664

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIDL2	CHP2	FAULT2[1:0]		IDLES2	IDLEM 2	POL2	Res.
								rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	BIAR	Res.	DLYPRT[2:0]			DLYPR TEN	DTEN	DIDL1	CHP1	FAULT1[1:0]		IDLES1	IDLEM 1	POL1	Res.
	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **DIDL2**: Output 2 deadtime upon burst mode Idle entry

This bit delays the idle mode entry by forcing a deadtime insertion before switching the outputs to their idle state. This setting only applies when entering in idle state during a burst mode operation.

0: The programmed Idle state is applied immediately to the output 2

1: Deadtime (inactive level) is inserted on output 2 before entering the idle mode. The deadtime value is set by DTFx[8:0].

Note: This parameter cannot be changed once the timer x is enabled.

DIDL=1 is set only if one of the outputs is active during the burst mode (IDLES=1), and with positive deadtimes (SDTR/SDTF set to 0).

Bit 22 **CHP2**: Output 2 chopper enable

This bit enables the chopper on output 2.

0: Output signal is not altered

1: Output signal is chopped by a carrier signal

Note: This parameter cannot be changed once the timer x is enabled.

Bits 21:20 **FAULT2[1:0]**: Output 2 fault state

These bits select the output 2 state after a fault event.

00: No action: the output is not affected by the fault input and stays in run mode.

01: Active

10: Inactive

11: High-Z

Note: This parameter cannot be changed once the timer x is enabled (TxCEN bit set), if FLTENx bit is set or if the output is in FAULT state.

Bit 19 **IDLES2**: Output 2 idle state

This bit selects the output 2 idle state.

0: Inactive

1: Active

Note: This parameter must be set prior to have the HRTIM controlling the outputs.

Bit 18 **IDLEM2**: Output 2 idle mode
This bit selects the output 2 idle mode.
0: No action: the output is not affected by the burst mode operation.
1: The output is in idle state when requested by the burst mode controller.
Note: This bit is preloaded and is changed during run-time, but must not be changed while the burst mode is active.

Bit 17 **POL2**: Output 2 polarity
This bit selects the output 2 polarity.
0: Positive polarity (output active high)
1: Negative polarity (output active low)
Note: This parameter cannot be changed once the timer x is enabled.

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **BIAR**: Balanced Idle Automatic Resume
This bit selects is the outputs are automatically re-enabled after a balanced idle event.
This bit is only significant if DLYPRT[2:0] = 011 or 111, it is ignored otherwise.
0: Disabled
1: Enabled
Note: This parameter cannot be changed once the timer x is enabled.

Bit 13 Reserved, must be kept at reset value.

Bits 12:10 **DLYPRT[2:0]**: Delayed protection
These bits define the source and outputs on which the delayed protection schemes are applied.
In HRTIM_OUTAR, HRTIM_OUTBR, HRTIM_OUTCR:
000: Output 1 delayed idle on external event F
001: Output 2 delayed idle on external event F
010: Output 1 and output 2 delayed idle on external event F
011: Balanced idle on external event F
100: Output 1 delayed idle on external event G
101: Output 2 delayed idle on external event G
110: Output 1 and output 2 delayed idle on external event G
111: Balanced idle on external event G
In HRTIM_OUTDR, HRTIM_OUTER, HRTIM_OUTFR:
000: Output 1 delayed idle on external event H
001: Output 2 delayed idle on external event H
010: Output 1 and output 2 delayed idle on external event H
011: Balanced idle on external event H
100: Output 1 delayed idle on external event I
101: Output 2 delayed idle on external event I
110: Output 1 and output 2 delayed idle on external event I
111: Balanced idle on external event I
Note: This bitfield must not be modified once the delayed protection is enabled (DLYPRTEN bit set).

Bit 9 **DLYPRTEN**: Delayed protection enable
This bit enables the delayed protection scheme.
0: No action
1: Delayed protection is enabled, as per DLYPRT[2:0] bits
Note: This parameter cannot be changed once the timer x is enabled (TxEN bit set).

Bit 8 **DTEN**: Deadtime enable

This bit enables the deadtime insertion on output 1 and output 2.

0: Output 1 and output 2 signals are independent.

1: Deadtime is inserted between output 1 and output 2 (reference signal is output 1 signal generator)

Note: This parameter cannot be changed if the timer outputs are enabled and set/reset by another timer.

Bit 7 **DIDL1**: Output 1 deadtime upon burst mode idle entry

This bit delays the idle mode entry by forcing a deadtime insertion before switching the outputs to their idle state. This setting only applies when entering the idle state during a burst mode operation.

0: The programmed idle state is applied immediately to the output 1

1: Deadtime (inactive level) is inserted on output 1 before entering the idle mode. The deadtime value is set by DTRx[8:0].

Note: This parameter cannot be changed once the timer x is enabled.

DIDL=1 is set only if one of the outputs is active during the burst mode (IDLES=1), and with positive deadtimes (SDTR/SDTF set to 0).

Bit 6 **CHP1**: Output 1 chopper enable

This bit enables the chopper on output 1.

0: Output signal is not altered

1: Output signal is chopped by a carrier signal

Note: This parameter cannot be changed once the timer x is enabled.

Bits 5:4 **FAULT1[1:0]**: Output 1 fault state

These bits select the output 1 state after a fault event

00: No action: the output is not affected by the fault input and stays in run mode.

01: Active

10: Inactive

11: High-Z

Note: This parameter cannot be changed once the timer x is enabled (TxCEN bit set), if FLTENx bit is set or if the output is in FAULT state.

Bit 3 **IDLES1**: Output 1 Idle State

This bit selects the output 1 idle state.

0: Inactive

1: Active

Note: This parameter must be set prior to HRTIM controlling the outputs.

Bit 2 **IDLEM1**: Output 1 Idle mode

This bit selects the output 1 idle mode.

0: No action: the output is not affected by the burst mode operation.

1: The output is in idle state when requested by the burst mode controller.

Note: This bit is preloaded and is changed during runtime, but must not be changed while burst mode is active.

Bit 1 **POL1**: Output 1 polarity

This bit selects the output 1 polarity.

0: Positive polarity (output active high)

1: Negative polarity (output active low)

Note: This parameter cannot be changed once the timer x is enabled.

Bit 0 Reserved, must be kept at reset value.

33.5.53 HRTIM timer x fault register (HRTIM_FLTxR) (x = A to F)

Address offset: Block A: 0x168

Address offset: Block B: 0x268

Address offset: Block C: 0x368

Address offset: Block D: 0x468

Address offset: Block E: 0x568

Address offset: Block F: 0x668

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTLCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT7A8EN	FLT6A7EN	FLT5A6EN	FLT4A5EN	FLT3A4EN	FLT2A3EN	FLT1A2EN
rwo									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8EN	FLT7EN	FLT6EN	FLT5EN	FLT4EN	FLT3EN	FLT2EN	FLT1EN
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLTLCK**: Fault sources lock
0: FLT1EN..FLT6EN bits are read/write
1: FLT1EN..FLT6EN bits are read only
The FLTLCK bit is write-once. Once it has been set, it cannot be modified till the next system reset.

Bits 30:23 Reserved, must be kept at reset value.

Bit 22 **FLT7A8EN**: Fault 7 AND Fault 8 enable
0: Fault 7 AND Fault 8 ignored
1: HRTIM outputs disabled when both Fault 7 AND Fault 8 inputs are active

Bit 21 **FLT6A7EN**: Fault 6 AND Fault 7 enable
0: Fault 6 AND Fault 7 ignored
1: HRTIM outputs disabled when both Fault 6 AND Fault 7 inputs are active

Bit 20 **FLT5A6EN**: Fault 5 AND Fault 6 enable
0: Fault 5 AND Fault 6 ignored
1: HRTIM outputs disabled when both Fault 5 AND Fault 6 inputs are active

Bit 19 **FLT4A5EN**: Fault 4 AND Fault 5 enable
0: Fault 4 AND Fault 5 ignored
1: HRTIM outputs disabled when both Fault 4 AND Fault 5 inputs are active

Bit 18 **FLT3A4EN**: Fault 3 AND Fault 4 enable
0: Fault 3 AND Fault 4 ignored
1: HRTIM outputs disabled when both Fault 3 AND Fault 4 inputs are active

Bit 17 **FLT2A3EN**: Fault 2 AND Fault 3 enable
0: Fault 2 AND Fault 3 ignored
1: HRTIM outputs disabled when both Fault 2 AND Fault 3 inputs are active

Bit 16 **FLT1A2EN**: Fault 1 AND Fault 2 enable
0: Fault 1 AND Fault 2 ignored
1: HRTIM outputs disabled when both Fault 1 AND Fault 2 inputs are active

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **FLT8EN**: Fault 8 enable
0: Fault 8 input ignored
1: Fault 8 input is active and disables HRTIM outputs

Bit 6 **FLT7EN**: Fault 7 enable
0: Fault 7 input ignored
1: Fault 7 input is active and disables HRTIM outputs

Bit 5 **FLT6EN**: Fault 6 enable
0: Fault 6 input ignored
1: Fault 6 input is active and disables HRTIM outputs

Bit 4 **FLT5EN**: Fault 5 enable
0: Fault 5 input ignored
1: Fault 5 input is active and disables HRTIM outputs

Bit 3 **FLT4EN**: Fault 4 enable
0: Fault 4 input ignored
1: Fault 4 input is active and disables HRTIM outputs

Bit 2 **FLT3EN**: Fault 3 enable
0: Fault 3 input ignored
1: Fault 3 input is active and disables HRTIM outputs

Bit 1 **FLT2EN**: Fault 2 enable
0: Fault 2 input ignored
1: Fault 2 input is active and disables HRTIM outputs

Bit 0 **FLT1EN**: Fault 1 enable
0: Fault 1 input ignored
1: Fault 1 input is active and disables HRTIM outputs

33.5.54 HRTIM timer x control register 2 (HRTIM_TIMxCR2) (x = A to F)

Address offset: Block A: 0x16C

Address offset: Block B: 0x26C

Address offset: Block C: 0x36C

Address offset: Block D: 0x46C

Address offset: Block E: 0x56C

Address offset: Block F: 0x66C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DCDCOE	COM	COR	COE	ADCTRGCH2	ADCTRGCH1	TRGHLFT	TRGHLF	Res.	Res.	GTCMP3	GTCMP1
				rw	rw	w	rw	rw	rw	rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEROM[1:0]		BMROM[1:0]		ADROM[1:0]		OUTROM[1:0]		ROM[1:0]		Res.	UDM	Res.	DCDR	DCDS	DCDE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **DCDCOE**: Dual channel DAC crossbar override enable

0: Dual Channel DAC Crossbar Override disabled

1: Dual Channel DAC Crossbar Override enabled

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 26 **COM**: Crossbar override mode

This bit defines if the crossbar override mode is entered only once (latched mode), in which case it is necessary to rearm it to resume standard crossbar operation, using the COR bit.

In cycle-by-cycle mode, the standard crossbar operation is automatically resumed on the next following reset/roll-over event, once the triggering condition have disappeared. This mode is only applicable in force on event mode (COFMx[2:0] = 100).

0: Cycle-by-cycle mode

1: latched mode

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 25 **COR**: Crossbar override rearm

This bit is set by software and automatically reset by hardware. It resume standard crossbar operation and rearms the crossbar override mode. The rearm operation is only effective if the crossbar override triggering condition have disappeared. If the trigger is still active, the rearm request is ignored and the crossbar override mode is maintained active.

Bit 24 **COE**: Crossbar override enable

0: Crossbar override disabled

1: Crossbar override enabled

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 23 **ADCTRGCH2**: ADC trigger for channel 2

This bit is significant only in push-pull mode. It gives the possibility to have ADC triggers issued only when Channel 2 output is active

0: ADC triggers generated for both Channel 1 and Channel 2 active phases

1: ADC triggers generated when Channel 2 is active (in push-pull mode)

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 22 **ADCTRGCH1**: ADC trigger for channel 1

This bit is significant only in push-pull mode. It gives the possibility to have ADC triggers issued only when Channel 1 output is active

0: ADC triggers generated for both Channel 1 and Channel 2 active phases

1: ADC triggers generated when Channel 1 is active (in push-pull mode)

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 21 **TRGHLFT**: Triggered-half timeout mode

This bit defines if a timeout event must be generated when the synchronization disappear (i.e. no more EEV event

0: No timeout event generated

1: timeout event generated. When the counter matches the PER value, the CMP2 register takes PER/2 value.

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 20 **TRGHLF**: Triggered-half mode

This bit defines whether the compare 2 register is behaving in standard mode (compare match issued as soon as counter equal compare), or in triggered-half mode (see *Section : Triggered-half mode*).

0: CMP2 register is written by the user only (standard compare mode)

1: CMP2 value is set by hardware as soon as a capture 1 event occurs. It is loaded with the (capture 1 divided by 2) value. The initial value can be written by the user (as long as TRGHLF is reset), but is ignored once the first capture has been triggered (the preload mechanism is disabled for CMP2 when the TRGHLF bit is set).

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **GTCMP3**: Greater than compare 3 PWM mode

This bit defines the compare 3 operating mode.

0: The compare 3 event is generated when the counter is equal to the compare value (compare match mode)

1: The compare 3 event is generated when the counter is greater than the compare value. If the compare value is changed on-the-fly, the new compare value is compared with the current counter value and an output SET or RESET can be generated.

Bit 16 **GTCMP1**: Greater than compare 1 PWM mode

This bit defines the compare 1 operating mode.

0: The compare 1 event is generated when the counter is equal to the compare value (compare match mode)

1: The compare 1 event is generated when the counter is greater than the compare value. If the compare value is changed on-the-fly, the new compare value is compared with the current counter value and an output SET or RESET can be generated.

Bits 15:14 **FEROM[1:0]**: Fault and event roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the Roll-over event used by the fault and event counters.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value

01: Event generated when the counter is equal to 0

10: Event generated when the counter is equal to HRTIM_PERxR

11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This bitfield cannot be changed once the timer is operating (TxEN bit set).

Bits 13:12 **BMROM[1:0]**: Burst mode roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the roll-over event used in the burst mode controller, as clock as burst mode trigger.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value

01: Event generated when the counter is equal to 0

10: Event generated when the counter is equal to HRTIM_PERxR

11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This parameter cannot be changed once the timer is operating (TxEN bit set).

Bits 11:10 **ADROM[1:0]**: ADC roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the roll-over event which triggers the ADC.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value

01: Event generated when the counter is equal to 0

10: Event generated when the counter is equal to HRTIM_PERxR

11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This bitfield cannot be changed once the timer is operating (TxEN bit set).

Bits 9:8 **OUTROM[1:0]**: Output roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the roll-over event which sets and/or resets the ouputs, as per HRTIM_SETxyR and HRTIM_RSTxyR settings.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value

01: Event generated when the counter is equal to 0

10: Event generated when the counter is equal to HRTIM_PERxR

11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This bitfield cannot be changed once the timer is operating (TxEN bit set).

Bits 7:6 **ROM[1:0]**: Roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the roll-over event with the following destinations: update trigger (to transfer content from preload to active registers), IRQ and DMA requests, repetition counter decrement and external event filtering.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value

01: Event generated when the counter is equal to 0

10: Event generated when the counter is equal to HRTIM_PERxR

11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This bitfield cannot be changed once the timer is operating (TxEN bit set).

Bit 5 Reserved, must be kept at reset value.

Bit 4 **UDM**: Up-Down mode
This bit defines if the counter is operating in up or up-down counting mode.
0: The counter is operating in up-counting mode
1: The counter is operating in up-down counting mode
Note: This bit cannot be changed once the timer is operating (TxEN bit set).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCDR**: Dual channel DAC reset trigger
This bit defines when the hrtim_dac_reset_trgx trigger is generated.
0: The trigger is generated on counter reset or roll-over event
1: The trigger is generated on output 1 set event
Note: The DCDR bit is not significant when the DCDE bit is reset (Dual channel DAC trigger disabled).

Bit 1 **DCDS**: Dual channel DAC Step trigger
This bit defines when the hrtim_dac_step_trgx trigger is generated.
0: The trigger is generated on compare 2 event
1: The trigger is generated on output 1 reset event
Note: The DCDR bit is not significant when the DCDE bit is reset (Dual channel DAC trigger disabled).

Bit 0 **DCDE**: Dual channel DAC trigger enable
This bit enables the dual channel DAC triggering mechanism.
0: Dual channel DAC trigger disabled
1: Dual channel DAC trigger enabled
Note: This bit cannot be changed once the timer is operating (TxEN bit set).

33.5.55 HRTIM timer x external event filtering register 3 (HRTIM_TIMxEEFR3) (x = A to F)

Address offset: Block A: 0x170

Address offset: Block B: 0x270

Address offset: Block C: 0x370

Address offset: Block D: 0x470

Address offset: Block E: 0x570

Address offset: Block F: 0x670

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EEVZCNT[5:0]						EEVZSEL[3:0]				Res.	EEVZRSTM	EEVZCRES	EEVZCE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	EEVYCNT[5:0]						EEVYSEL[3:0]				Res.	EEVYRSTM	EEVYCRES	EEVYCE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **EEVZCNT[5:0]**: External event Z counter

This bitfield selects the external event Z counter threshold. An event is considered valid when the number of events is equal to the (EEVZCNT[5:0]+1) value.

Bits 23:20 **EEVZSEL[3:0]**: External event Z selection

This bit selects the external event B source.

0: External event A is used as external event Z source

1: External event B is used as external event Z source

...

9: External event J is used as external event Z source

Others: Reserved

Bit 19 Reserved, must be kept at reset value.

Bit 18 **EEVZRSTM**: External event Z reset mode

This bit selects the external event Z counter reset mode

0: External event counter Z is reset on each reset / roll-over event

1: External event counter Z is reset on each reset / roll-over event only if no event occurs during last counting period

Bit 17 **EEVZCRES**: External event Z counter reset

This bit resets the external event Z counter. It is set by software and reset by hardware.

0: No action

1: External event counter Z counter is reset

Bit 16 **EEVZCE**: External event Z counter enable

This bit enables the external event Z counter.

0: External event Z counter disabled

1: External event Z counter enabled

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

Bits 13:8 **EEVYCNT[5:0]**: External event Y counter
This bitfield selects the external event Y counter threshold. An event is considered valid when the number of events is equal to the (EEVYCNT[5:0]+1) value.

Bits 7:4 **EEVYSEL[3:0]**: External event Y Selection
This bit selects the external event Y source.
0: External event A is used as external event Y source
1: External event B is used as external event Y source
...
9: External event J is used as external event Y source
Others: Reserved

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EEVYRSTM**: External event Y reset mode
This bit selects the external event Y counter reset mode.
0: External event counter Y is reset on each reset / roll-over event
1: External event counter Y is reset on each reset / roll-over event only if no event occurs during last counting period

Bit 1 **EEVYCRES**: External event Y counter reset
This bit resets the external event Y counter. It is set by software and reset by hardware.
0: No action
1: External event counter Y is reset

Bit 0 **EEVYCE**: External event Y counter enable
This bit enables the external event Y counter.
0: External event Y counter disabled
1: External event Y counter enabled

33.5.56 HRTIM timer x external event selection register (HRTIM_EESELxR) (x = A to F)

Address offset: Block A: 0x174

Address offset: Block B: 0x274

Address offset: Block C: 0x374

Address offset: Block D: 0x474

Address offset: Block E: 0x574

Address offset: Block F: 0x674

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EEVJSEL[1:0]		Res.	EEVISEL[1:0]		Res.	EEVHSEL[1:0]		Res.	EEVGSEL[1:0]		Res.	EEVFSEL[1:0]	
		rw	rw		rw	rw		rw	rw		rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	EEVESEL[1:0]		Res.	EEVDSEL[1:0]		Res.	EEVCSEL[1:0]		Res.	EEVBSEL[1:0]		Res.	EEVASEL[1:0]	
		rw	rw		rw	rw		rw	rw		rw	rw		rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **EEVJSEL[1:0]**: External event J selection
00: EEVJ is connected to EEV10
01: EEVJ is connected to EEV14
10: EEVJ is connected to (EEV10 AND EEV11)
11: EEVJ is connected to (EEV10 AND EEV14)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 27 Reserved, must be kept at reset value.

Bits 26:25 **EEVISEL[1:0]**: External event I selection
00: EEVI is connected to EEV9
01: EEVI is connected to EEV13
10: EEVI is connected to (EEV9 AND EEV10)
11: EEVI is connected to (EEV9 AND EEV13)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 24 Reserved, must be kept at reset value.

Bits 23:22 **EEVHSEL[1:0]**: External event H selection
00: EEVH is connected to EEV8
01: EEVH is connected to EEV12
10: EEVH is connected to (EEV8 AND EEV9)
11: EEVH is connected to (EEV8 AND EEV12)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 21 Reserved, must be kept at reset value.

Bits 20:19 **EEVGSEL[1:0]**: External event G selection
00: EEVG is connected to EEV7
01: EEVG is connected to EEV11
10: EEVG is connected to (EEV7 AND EEV8)
11: EEVG is connected to (EEV7 AND EEV11)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **EEVFSEL[1:0]**: External event F selection
00: EEVF is connected to EEV6
01: EEVF is connected to EEV16
10: EEVF is connected to (EEV6 AND EEV7)
11: EEVF is connected to (EEV6 AND EEV16)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

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

Bits 13:12 **EEVESEL[1:0]**: External event E selection
00: EEVE is connected to EEV5
01: EEVE is connected to EEV15
10: EEVE is connected to (EEV5 AND EEV6)
11: EEVE is connected to (EEV5 AND EEV15)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 11 Reserved, must be kept at reset value.

Bits 10:9 **EEVDSEL[1:0]**: External event D selection
00: EEVD is connected to EEV4
01: EEVD is connected to EEV14
10: EEVD is connected to (EEV4 AND EEV5)
11: EEVD is connected to (EEV4 AND EEV14)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **EEVCSEL[1:0]**: External event C selection
00: EEVC is connected to EEV3
01: EEVC is connected to EEV13
10: EEVC is connected to (EEV3 AND EEV4)
11: EEVC is connected to (EEV3 AND EEV13)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **EEVBSEL[1:0]**: External event B selection
00: EEVB is connected to EEV2
01: EEVB is connected to EEV12
10: EEVB is connected to (EEV2 AND EEV3)
11: EEVB is connected to (EEV2 AND EEV12)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **EEVASEL[1:0]**: External event A selection
00: EEVA is connected to EEV1
01: EEVA is connected to EEV11
10: EEVA is connected to (EEV1 AND EEV2)
11: EEVA is connected to (EEV1 AND EEV11)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

33.5.57 HRTIM timer x crossbar override trigger register (HRTIM_COTRGxR) (x = A to F)

Address offset: Block A: 0x178

Address offset: Block B: 0x278

Address offset: Block C: 0x378

Address offset: Block D: 0x478

Address offset: Block E: 0x578

Address offset: Block F: 0x678

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXT_TRG5	EXT_TRG4	EXT_TRG3	EXT_TRG2	EXT_TRG1	EXT_TRG0	EEVJ	EEVI	EEVH	EEVG	EEVF	EEVE	EEVD	EEVC	EEVB	EEVA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT7A8	FLT6A7	FLT5A6	FLT4A5	FLT3A4	FLT2A3	FLT1A2	FLT8	FLT7	FLT6	FLT5	FLT4	FLT3	FLT2	FLT1	SWT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rs

Bit 31 **EXT_TRG5**: External trigger 5
0: External trigger 5 ignored
1: External trigger 5 enabled

Bit 30 **EXT_TRG4**: External trigger 4
0: External trigger 4 ignored
1: External trigger 4 enabled

Bit 29 **EXT_TRG3**: External trigger 3
0: External trigger 3 ignored
1: External trigger 3 enabled

Bit 28 **EXT_TRG2**: External trigger 2
0: External trigger 2 ignored
1: External trigger 2 enabled

Bit 27 **EXT_TRG1**: External trigger 1
0: External trigger 1 ignored
1: External trigger 1 enabled

Bit 26 **EXT_TRG0**: External trigger 0
0: External trigger 0 ignored
1: External trigger 0 enabled

Bit 25 **EEVJ**: EEVJ trigger
0: EEVJ input trigger ignored
1: EEVJ input trigger enabled

Bit 24 **EEVI**: EEVI trigger
0: EEVI input trigger ignored
1: EEVI input trigger enabled

Bit 23 **EEVH**: EEVH trigger
0: EEVH input trigger ignored
1: EEVH input trigger enabled

Bit 22 **EEVG**: EEVG trigger
0: EEVG input trigger ignored
1: EEVG input trigger enabled

Bit 21 **EEVF**: EEVF trigger
0: EEVF input trigger ignored
1: EEVF input trigger enabled

Bit 20 **EEVE**: EEVE trigger
0: EEVE input trigger ignored
1: EEVE input trigger enabled

Bit 19 **EEVD**: EEVD trigger
0: EEVD input trigger ignored
1: EEVD input trigger enabled

Bit 18 **EEVC**: EEVC trigger
0: EEVC input trigger ignored
1: EEVC input trigger enabled

Bit 17 **EEVB**: EEVB trigger
0: EEVB input trigger ignored
1: EEVB input trigger enabled

Bit 16 **EEVA**: EEVA trigger
0: EEVA input trigger ignored
1: EEVA input trigger enabled

Bit 15 **FLT7A8**: FLT7A8 trigger
0: (Fault 7 AND Fault 8) trigger ignored
1: (Fault 7 AND Fault 8) trigger enabled

Bit 14 **FLT6A7**: FLT6A7 trigger
0: (Fault 6 AND Fault 7) trigger ignored
1: (Fault 6 AND Fault 7) trigger enabled

Bit 13 **FLT5A6**: FLT5A6 trigger
0: (Fault 5 AND Fault 6) trigger ignored
1: (Fault 5 AND Fault 6) trigger enabled

Bit 12 **FLT4A5**: FLT4A5 trigger
0: (Fault 4 AND Fault 5) trigger ignored
1: (Fault 4 AND Fault 5) trigger enabled

Bit 11 **FLT3A4**: FLT2A3 trigger
0: (Fault 3 AND Fault 4) trigger ignored
1: (Fault 3 AND Fault 4) trigger enabled

Bit 10 **FLT2A3**: FLT2A3 trigger
0: (Fault 2 AND Fault 3) trigger ignored
1: (Fault 2 AND Fault 3) trigger enabled

Bit 9 **FLT1A2**: FLT1A2 trigger
0: (Fault 1 AND Fault 2) trigger ignored
1: (Fault 1 AND Fault 2) trigger enabled

Bit 8 **FLT8**: FLT8 trigger
0: Fault 8 input trigger ignored
1: Fault 8 input trigger enabled

Bit 7 **FLT7**: FLT7 trigger
0: Fault 7 input trigger ignored
1: Fault 7 input trigger enabled

Bit 6 **FLT6**: FLT6 trigger
0: Fault 6 input trigger ignored
1: Fault 6 input trigger enabled

Bit 5 **FLT5**: FLT5 trigger
0: Fault 5 input trigger ignored
1: Fault 5 input trigger enabled

Bit 4 **FLT4**: FLT4 trigger
0: Fault 4 input trigger ignored
1: Fault 4 input trigger enabled

Bit 3 **FLT3**: FLT3 trigger
0: Fault 3 input trigger ignored
1: Fault 3 input trigger enabled

Bit 2 **FLT2**: FLT2 trigger
0: Fault 2 input trigger ignored
1: Fault 2 input trigger enabled

Bit 1 **FLT1**: FLT1 trigger
0: Fault 1 input trigger ignored
1: Fault 1 input trigger enabled

Bit 0 **SWT**: Software trigger
0: No action
1: Crossbar override software trigger

33.5.58 HRTIM timer x crossbar override forcing event register (HRTIM_COFExR) (x = A to F)

Address offset: Block A: 0x17C

Address offset: Block B: 0x27C

Address offset: Block C: 0x37C

Address offset: Block D: 0x47C

Address offset: Block E: 0x57C

Address offset: Block F: 0x67C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EEVJFRCB	EEVIFRCB	EEVHFRCB	EEVGFRCB	EEVFFRCB	EEVEFRCB	EEVDFRCB	EEVCFRCB	EEVBFRCB	EEVAFRCB	Res.	COFMB[2:0]		
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	EEVJFRCA	EEVIFRCA	EEVHFRCA	EEVGFRCA	EEVFFRCA	EEVEFRCA	EEVDFRCA	EEVCFRCA	EEVBFRCA	EEVAFRCA	Res.	COFMA[2:0]		
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **EEVJFRCB**: External event J forces output B
0: No action
1: Output B is forced to the active level when External Event J is active

Bit 28 **EEVIFRCB**: External event I forces output B
0: No action
1: Output B is forced to the active level when External Event I is active

Bit 27 **EEVHFRCB**: External event H forces output B
0: No action
1: Output B is forced to the active level when External Event H is active

Bit 26 **EEVGFRCB**: External event G forces output B
0: No action
1: Output B is forced to the active level when External Event G is active

Bit 25 **EEVFFRCB**: External event F forces output B
0: No action
1: Output B is forced to the active level when External Event F is active

Bit 24 **EEVEFRCB**: External event E forces output B
0: No action
1: Output B is forced to active level when External Event E is active

Bit 23 **EEVDFRCB**: External event D forces output B
0: No action
1: Output B is forced to the active level when External Event D is active

Bit 22 **EEVCFRCB**: External event C forces output B
0: No action
1: Output B is forced to the active level when External Event C is active

Bit 21 **EEVBFRCB**: External event B forces output B
0: No action
1: Output B is forced to the active level when External Event B is active

Bit 20 **EEVAFRCB**: External event A forces output B
0: No action
1: Output B is forced to active level when External Event A is active

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **COFMB[2:0]**: Crossbar override force mode output B
This bitfield defines how the output B is handled once the crossbar override mode is triggered
000: Delayed 1 fault
Output B is forced to the inactive level after Delay 1 duration defined with DLY1FLTD[8:0] bitfield in HRTIM_CODExR register.
001: Delayed 2 fault
Output B is forced to the inactive level after Delay 2 duration defined with DLY2FLTD[8:0] bitfield in HRTIM_CODExR register
Others: reserved

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

Bit 13 **EEVJFRCA**: External event J forces output A
0: No action
1: Output A is forced to the active level when External Event J is active

Bit 12 **EEVIFRCA**: External event I forces output A
0: No action
1: Output A is forced to the active level when External Event I is active

Bit 11 **EEVHFRCA**: External event H forces output A
0: No action
1: Output A is forced to the active level when External Event H is active

Bit 10 **EEVGFRCA**: External event G forces output A
0: No action
1: Output A is forced to the active level when External Event G is active

Bit 9 **EEVFFRCA**: External event F forces output A
0: No action
1: Output A is forced to the active level when External Event F is active

Bit 8 **EEVEFRCA**: External event E forces output A
0: No action
1: Output A is forced to active level when External Event E is active

Bit 7 **EEVDFRCA**: External event D forces output A
0: No action
1: Output A is forced to the active level when External Event D is active

Bit 6 **EEVCFRCA**: External event C forces output A
0: No action
1: Output A is forced to the active level when External Event C is active

Bit 5 **EEVBFRCA**: External event B forces output A
0: No action
1: Output A is forced to the active level when External Event B is active

Bit 4 **EEVAFRCA**: External event A forces output A
0: No action
1: Output A is forced to active level when External Event A is active

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **COFMA[2:0]**: Crossbar override force mode output A
This bitfield defines how the output A is handled once the crossbar override mode is triggered
000: Delayed 1 fault
Output A is forced to the inactive level after Delay 1 duration defined with DLY1FLTD[8:0] bitfield in HRTIM_CODExR register.
001: Delayed 2 fault
Output A is forced to the inactive level after Delay 2 duration defined with DLY2FLTD[8:0] bitfield in HRTIM_CODExR register
Others: reserved

33.5.59 HRTIM timer x crossbar override delayed fault event register (HRTIM_CODExR) (x = A to F)

Address offset: Block A: 0x180

Address offset: Block B: 0x280

Address offset: Block C: 0x380

Address offset: Block D: 0x480

Address offset: Block E: 0x580

Address offset: Block F: 0x680

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLY2FLTD[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLY1FLTD[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 **DLY2FLTD[8:0]**: Delayed 2 Fault duration

This register holds the value of the delay applied before entering in Fault mode, for Delayed 1 fault mode.

t_{Delay2} = DLY2FLTD[8:0] x t_{DTG}

Bits 15:9 Reserved, must be kept at reset value.

Bits 8:0 **DLY1FLTD[8:0]**: Delayed 1 Fault duration

This register holds the value of the delay applied before entering in Fault mode, for Delayed 1 fault mode.

t_{Delay1} = DLY1FLTD[8:0] x t_{DTG}

33.5.60 HRTIM control register 1 (HRTIM_CR1)

Address offset: 0x700

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	AD4USRC[2:0]			AD3USRC[2:0]			AD2USRC[2:0]			AD1USRC[2:0]		
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RTFUDIS	RTEUDIS	RTDUDIS	RTCUDIS	RTBUDIS	RTAUDIS	RMUDIS	Res.	TFUDIS	TEUDIS	TDUDIS	TCUDIS	TBUDIS	TAUDIS	MUDIS
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:25 **AD4USRC[2:0]**: ADC trigger 4 update source
Refer to AD1USRC[2:0] description.

Bits 24:22 **AD3USRC[2:0]**: ADC trigger 3 update source
Refer to AD1USRC[2:0] description.

Bits 21:19 **AD2USRC[2:0]**: ADC trigger 2 update source
Refer to AD1USRC[2:0] description.

Bits 18:16 **AD1USRC[2:0]**: ADC trigger 1 update source
These bits define the source which triggers the update of the HRTIM_ADC1R register (transfer from preload to active register). It only defines the source timer. The precise condition is defined within the timer itself, in HRTIM_MCR or HRTIM_TIMxCR.
000: Master timer
001: Timer A
010: Timer B
011: Timer C
100: Timer D
101: Timer E
110: Timer F
111: Reserved

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RTFUDIS**: Remote Timer F update disable
Refer to RTAUDIS description.

Bit 13 **RTEUDIS**: Remote Timer E update disable
Refer to RTAUDIS description

Bit 12 **RTDUDIS**: Remote Timer D update disable
Refer to RTAUDIS description.

Bit 11 **RTCUDIS**: Remote Timer C update disable
Refer to RTAUDIS description.

Bit 10 **RTBUDIS**: Remote Timer B update disable
Refer to RTAUDIS description.

Bit 9 **RTAUDIS**: Remote timer A update disable

This bit is set and cleared by software to enable/disable an update event generation temporarily in another HRTIMER's timer A.

0: Update enabled. The update occurs upon generation of the selected source.

1: Update disabled. The updates are temporarily disabled to allow the software to write multiple registers that have to be simultaneously taken into account.

Bit 8 **RMUDIS**: Remote master update disable

This bit is set and cleared by software to enable/disable an update event generation temporarily in another HRTIMER's master timer

0: Update enabled.

1: Update disabled. The updates are temporarily disabled to allow the software to write multiple registers that have to be simultaneously taken into account.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TFUDIS**: Timer F update disable

Refer to TAUDIS description.

Bit 5 **TEUDIS**: Timer E update disable

Refer to TAUDIS description

Bit 4 **TDUDIS**: Timer D update disable

Refer to TAUDIS description.

Bit 3 **TCUDIS**: Timer C update disable

Refer to TAUDIS description.

Bit 2 **TBUDIS**: Timer B update disable

Refer to TAUDIS description.

Bit 1 **TAUDIS**: Timer A update disable

This bit is set and cleared by software to enable/disable an update event generation temporarily on timer A.

0: Update enabled. The update occurs upon generation of the selected source.

1: Update disabled. The updates are temporarily disabled to allow the software to write multiple registers that have to be simultaneously taken into account.

Bit 0 **MUDIS**: Master update disable

This bit is set and cleared by software to enable/disable an update event generation temporarily.

0: Update enabled.

1: Update disabled. The updates are temporarily disabled to allow the software to write multiple registers that have to be simultaneously taken into account.

33.5.61 HRTIM control register 2 (HRTIM_CR2)

Address offset: 0x704

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	DBSS[2:0]			Res.	Res.	SWPF	SWPE	SWPD	SWPC	SWPB	SWPA
					rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TFRST	TERST	TDRST	TCRST	TBRST	TARST	MRST	Res.	TFSWU	TESWU	TDSWU	TCSWU	TBSWU	TASWU	MSWU
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

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

Bits 26:24 DBSS[2:0]: DMA burst source selection

This bitfield defines the DMA source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done at the HRTIM_BDMADR register address).

000:Reserved
001:Master timer
010:Timer A
011:Timer B
100:Timer C
101:Timer D
110:Timer E
111:Timer F

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **SWPF**: Swap timer F outputs

Refer to SWPA description.

Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 20 **SWPE**: Swap timer E outputs

Refer to SWPA description.

Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 19 **SWPD**: Swap timer D outputs

Refer to SWPA description.

Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 18 **SWPC**: Swap timer C outputs

Refer to SWPA description.

Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 17 **SWPB**: Swap timer B outputs

Refer to SWPA description.

Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 16 **SWPA**: Swap timer A outputs
This bit allows to swap the timer A outputs.
0: HRTIM_SETA1R and HRTIM_RSTA1R are coding for the output A1, HRTIM_SETA2R and HRTIM_RSTA2R are coding for the output A2
1: HRTIM_SETA1R and HRTIM_RSTA1R are coding for the output A2, HRTIM_SETA2R and HRTIM_RSTA2R are coding for the output A1
Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TFRST**: Timer F counter software reset
Refer to TARST description.

Bit 13 **TERST**: Timer E counter software reset
Refer to TARST description.

Bit 12 **TDRST**: Timer D counter software reset
Refer to TARST description.

Bit 11 **TCRST**: Timer C counter software reset
Refer to TARST description.

Bit 10 **TBRST**: Timer B counter software reset
Refer to TARST description.

Bit 9 **TARST**: Timer A counter software reset
Setting this bit resets the timer A counter.
The bit is automatically reset by hardware.

Bit 8 **MRST**: Master counter software reset
Setting this bit resets the master timer counter.
The bit is automatically reset by hardware.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TFSWU**: Timer F software update
Refer to TASWU description.

Bit 5 **TESWU**: Timer E software update
Refer to TASWU description.

Bit 4 **TDSWU**: Timer D software update
Refer to TASWU description.

Bit 3 **TCSWU**: Timer C software update
Refer to TASWU description.

Bit 2 **TBSWU**: Timer B software update
Refer to TASWU description.

Bit 1 **TASWU**: Timer A software update
This bit is set by software and automatically reset by hardware. It forces an immediate transfer from the preload to the active register and any pending update request is canceled.

Bit 0 **MSWU**: Master timer software update
This bit is set by software and automatically reset by hardware. It forces an immediate transfer from the preload to the active register in the master timer and any pending update request is canceled.

33.5.62 HRTIM interrupt status register (HRTIM_ISR)

Address offset: 0x708

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPER	Res.
														r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8	FLT7	FLT6	SYSFLT	FLT5	FLT4	FLT3	FLT2	FLT1
							r	r	r	r	r	r	r	r	r

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

Bit 17 **BMPER**: Burst mode period interrupt flag
This bit is set by hardware when a single-shot burst mode operation is completed or at the end of a burst mode period in continuous mode. It is cleared by software writing it at 1.
0: No burst mode period interrupt occurred
1: Burst mode period interrupt occurred

Bits 16:9 Reserved, must be kept at reset value.

Bit 8 **FLT8**: Fault 8 interrupt flag
Refer to FLT1 description.

Bit 7 **FLT7**: Fault 7 interrupt flag
Refer to FLT1 description.

Bit 6 **FLT6**: Fault 6 interrupt flag
Refer to FLT1 description.

Bit 5 **SYSFLT**: System fault interrupt flag
Refer to FLT1 description.

Bit 4 **FLT5**: Fault 5 interrupt flag
Refer to FLT1 description.

Bit 3 **FLT4**: Fault 4 interrupt flag
Refer to FLT1 description.

Bit 2 **FLT3**: Fault 3 interrupt flag
Refer to FLT1 description.

Bit 1 **FLT2**: Fault 2 interrupt flag
Refer to FLT1 description.

Bit 0 **FLT1**: Fault 1 interrupt flag
This bit is set by hardware when fault 1 event occurs. It is cleared by software writing it at 1.
0: No fault 1 interrupt occurred
1: Fault 1 interrupt occurred

33.5.63 HRTIM interrupt clear register (HRTIM_ICR)

Address offset: 0x70C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERC	Res.
														w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8C	FLT7C	FLT6C	SYSFLTC	FLT5C	FLT4C	FLT3C	FLT2C	FLT1C
							w	w	w	w	w	w	w	w	w

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

Bit 17 **BMPERC**: Burst mode period flag clear
Writing 1 to this bit clears the BMPER flag in HRTIM_ISR register.

Bits 16:9 Reserved, must be kept at reset value.

Bit 8 **FLT8C**: Fault 8 interrupt flag clear
Writing 1 to this bit clears the FLT6 flag in HRTIM_ISR register.

Bit 7 **FLT7C**: Fault 7 interrupt flag clear
Writing 1 to this bit clears the FLT6 flag in HRTIM_ISR register.

Bit 6 **FLT6C**: Fault 6 interrupt flag clear
Writing 1 to this bit clears the FLT6 flag in HRTIM_ISR register.

Bit 5 **SYSFLTC**: System fault interrupt flag clear
Writing 1 to this bit clears the SYSFLT flag in HRTIM_ISR register.

Bit 4 **FLT5C**: Fault 5 interrupt flag clear
Writing 1 to this bit clears the FLT5 flag in HRTIM_ISR register.

Bit 3 **FLT4C**: Fault 4 interrupt flag clear
Writing 1 to this bit clears the FLT4 flag in HRTIM_ISR register.

Bit 2 **FLT3C**: Fault 3 interrupt flag clear
Writing 1 to this bit clears the FLT3 flag in HRTIM_ISR register.

Bit 1 **FLT2C**: Fault 2 interrupt flag clear
Writing 1 to this bit clears the FLT2 flag in HRTIM_ISR register.

Bit 0 **FLT1C**: Fault 1 interrupt flag clear
Writing 1 to this bit clears the FLT1 flag in HRTIM_ISR register.

33.5.64 HRTIM interrupt enable register (HRTIM_IER)

Address offset: 0x710

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERIE	Res.
														rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8IE	FLT7IE	FLT6IE	SYSFLTIE	FLT5IE	FLT4IE	FLT3IE	FLT2IE	FLT1IE
							rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 17 **BMPERIE**: Burst mode period interrupt enable
This bit is set and cleared by software to enable/disable the burst mode period interrupt.
0: Burst mode period interrupt disabled
1: Burst mode period interrupt enabled

Bits 16:9 Reserved, must be kept at reset value.

Bit 8 **FLT8IE**: Fault 8 interrupt enable
Refer to FLT1IE description.

Bit 7 **FLT7IE**: Fault 7 interrupt enable
Refer to FLT1IE description.

Bit 6 **FLT6IE**: Fault 6 interrupt enable
Refer to FLT1IE description.

Bit 5 **SYSFLTIE**: System fault interrupt enable
Refer to FLT1IE description.

Bit 4 **FLT5IE**: Fault 5 interrupt enable
Refer to FLT1IE description.

Bit 3 **FLT4IE**: Fault 4 interrupt enable
Refer to FLT1IE description.

Bit 2 **FLT3IE**: Fault 3 interrupt enable
Refer to FLT1IE description.

Bit 1 **FLT2IE**: Fault 2 interrupt enable
Refer to FLT1IE description.

Bit 0 **FLT1IE**: Fault 1 interrupt enable
This bit is set and cleared by software to enable/disable the fault 1 interrupt.
0: Fault 1 interrupt disabled
1: Fault 1 interrupt enabled

33.5.65 HRTIM output enable register (HRTIM_OENR)

Address offset: 0x714

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TF2OEN	TF1OEN	TE2OEN	TE1OEN	TD2OEN	TD1OEN	TC2OEN	TC1OEN	TB2OEN	TB1OEN	TA2OEN	TA1OEN
				rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **TF2OEN**: Timer F output 2 enable
Refer to TA1OEN description.

Bit 10 **TF1OEN**: Timer F output 1 enable
Refer to TA1OEN description

Bit 9 **TE2OEN**: Timer E output 2 enable
Refer to TA1OEN description.

Bit 8 **TE1OEN**: Timer E output 1 enable
Refer to TA1OEN description.

Bit 7 **TD2OEN**: Timer D output 2 enable
Refer to TA1OEN description.

Bit 6 **TD1OEN**: Timer D output 1 enable
Refer to TA1OEN description.

Bit 5 **TC2OEN**: Timer C output 2 enable
Refer to TA1OEN description.

Bit 4 **TC1OEN**: Timer C output 1 enable
Refer to TA1OEN description.

Bit 3 **TB2OEN**: Timer B output 2 enable
Refer to TA1OEN description.

Bit 2 **TB1OEN**: Timer B output 1 enable
Refer to TA1OEN description.

Bit 1 **TA2OEN**: Timer A output 2 enable
Refer to TA1OEN description.

Bit 0 **TA1OEN**: Timer A output 1 enable
Setting this bit enables the timer A output 1. Writing “0” has no effect.
Reading the bit returns the output enable/disable status.
This bit is cleared asynchronously by hardware as soon as the timer-related fault input(s) is (are) active.
0: output A1 disabled. The output is either in fault or Idle state.
1: output A1 enabled

Note: The disable status corresponds to both idle and fault states. The output disable status is given by TA1ODS bit in the HRTIM_ODSR register.

33.5.66 HRTIM output disable register (HRTIM_ODISR)

Address offset: 0x718

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TF2ODIS	TF1ODIS	TE2ODIS	TE1ODIS	TD2ODIS	TD1ODIS	TC2ODIS	TC1ODIS	TB2ODIS	TB1ODIS	TA2ODIS	TA1ODIS
				w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **TF2ODIS**: Timer F output 2 disable
Refer to TA1ODIS description.

Bit 10 **TF1ODIS**: Timer F output 1 disable
Refer to TA1ODIS description

Bit 9 **TE2ODIS**: Timer E output 2 disable
Refer to TA1ODIS description.

Bit 8 **TE1ODIS**: Timer E output 1 disable
Refer to TA1ODIS description.

Bit 7 **TD2ODIS**: Timer D output 2 disable
Refer to TA1ODIS description.

Bit 6 **TD1ODIS**: Timer D output 1 disable
Refer to TA1ODIS description.

Bit 5 **TC2ODIS**: Timer C output 2 disable
Refer to TA1ODIS description.

Bit 4 **TC1ODIS**: Timer C output 1 disable
Refer to TA1ODIS description.

Bit 3 **TB2ODIS**: Timer B output 2 disable
Refer to TA1ODIS description.

Bit 2 **TB1ODIS**: Timer B output 1 disable
Refer to TA1ODIS description.

Bit 1 **TA2ODIS**: Timer A output 2 disable
Refer to TA1ODIS description.

Bit 0 **TA1ODIS**: Timer A output 1 disable
Setting this bit disables the timer A output 1. The output enters the idle state, either from the run state or from the fault state.
Writing “0” has no effect.

33.5.67 HRTIM output disable status register (HRTIM_ODSR)

Address offset: 0x71C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TF2ODS	TF1ODS	TE2ODS	TE1ODS	TD2ODS	TD1ODS	TC2ODS	TC1ODS	TB2ODS	TB1ODS	TA2ODS	TA1ODS
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **TF2ODS**: Timer F output 2 disable status
Refer to TA1ODS description.

Bit 10 **TF1ODS**: Timer F output 1 disable status
Refer to TA1ODS description.

Bit 9 **TE2ODS**: Timer E output 2 disable status
Refer to TA1ODS description.

Bit 8 **TE1ODS**: Timer E output 1 disable status
Refer to TA1ODS description.

Bit 7 **TD2ODS**: Timer D output 2 disable status
Refer to TA1ODS description.

Bit 6 **TD1ODS**: Timer D output 1 disable status
Refer to TA1ODS description.

Bit 5 **TC2ODS**: Timer C output 2 disable status
Refer to TA1ODS description.

Bit 4 **TC1ODS**: Timer C output 1 disable status
Refer to TA1ODS description.

Bit 3 **TB2ODS**: Timer B output 2 disable status
Refer to TA1ODS description.

Bit 2 **TB1ODS**: Timer B output 1 disable status
Refer to TA1ODS description.

Bit 1 **TA2ODS**: Timer A output 2 disable status
Refer to TA1ODS description.

Bit 0 **TA1ODS**: Timer A output 1 disable status
Reading the bit returns the output disable status. It is not significant when the output is active (Tx1OEN or Tx2OEN = 1).
0: Output A1 disabled, in Idle state.
1: Output A1 disabled, in fault state.

33.5.68 HRTIM burst mode control register (HRTIM_BMCR)

Address offset: 0x720

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BMSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFBM	TEBM	TDBM	TCBM	TBBM	TABM	MTBM
rc_w0									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BMPREN	BMPRSC[3:0]				BMCLK[3:0]				BMOM	BME
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BMSTAT**: Burst mode status

This bit gives the current operating state.

0: Normal operation

1: Burst operation on-going. Writing this bit to 0 causes a burst mode early termination.

Bits 30:23 Reserved, must be kept at reset value.

Bit 22 **TFBM**: Timer F burst mode

Refer to TABM description.

Bit 21 **TEBM**: Timer E burst mode

Refer to TABM description.

Bit 20 **TDBM**: Timer D burst mode

Refer to TABM description.

Bit 19 **TCBM**: Timer C burst mode

Refer to TABM description.

Bit 18 **TBBM**: Timer B burst mode

Refer to TABM description.

Bit 17 **TABM**: Timer A burst mode

This bit defines how the timer behaves during a burst mode operation. This bitfield cannot be changed while the burst mode is enabled.

0: TA counter clock is maintained and the timer operates normally

1: TA counter clock is stopped and the counter is reset

Note: This bit must not be set when the balanced idle mode is active (DLYPRT[2:0] = 0x11).

Bit 16 **MTBM**: Master timer burst mode

This bit defines how the timer behaves during a burst mode operation. This bitfield cannot be changed while the burst mode is enabled.

0: Master Timer counter clock is maintained and the timer operates normally

1: Master Timer counter clock is stopped and the counter is reset

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **BMPREN**: Burst mode preload enable

This bit enables the registers preload mechanism and defines whether a write access into a preloadable register (HRTIM_BMCMPR, HRTIM_BMPER) is done into the active or the preload register.

0: Preload disabled: the write access is directly done into active registers

1: Preload enabled: the write access is done into preload registers

Bits 9:6 **BMPRSC[3:0]**: Burst mode prescaler

Defines the prescaling ratio of the f_{HRTIM} clock for the burst mode controller. This bitfield cannot be changed while the burst mode is enabled.

0000: Clock not divided
0001: Division by 2
0010: Division by 4
0011: Division by 8
0100: Division by 16
0101: Division by 32
0110: Division by 64
0111: Division by 128
1000: Division by 256
1001: Division by 512
1010: Division by 1024
1011: Division by 2048
1100: Division by 4096
1101: Division by 8192
1110: Division by 16384
1111: Division by 32768

Bits 5:2 **BMCLK[3:0]**: Burst mode clock source

This bitfield defines the clock source for the burst mode counter. It cannot be changed while the burst mode is enabled (refer to *Table 328* for on-chip events 1..4 connections details).

0000: Master timer counter reset/roll-over
0001: Timer A counter reset/roll-over
0010: Timer B counter reset/roll-over
0011: Timer C counter reset/roll-over
0100: Timer D counter reset/roll-over
0101: Timer E counter reset/roll-over
0110: On-chip event 1 (hrtim_bm_ck1), acting as a burst mode counter clock
0111: On-chip event 2 (hrtim_bm_ck2) acting as a burst mode counter clock
1000: On-chip event 3 (hrtim_bm_ck3) acting as a burst mode counter clock
1001: On-chip event 4 (hrtim_bm_ck4) acting as a burst mode counter clock
1010: Prescaled f_{HRTIM} clock (as per BMPRSC[3:0] setting)
1011: Timer F counter reset/roll-over
Others: Reserved

Bit 1 **BMOM**: Burst mode operating mode

This bit defines if the burst mode is entered once or if it is continuously operating.

0: Single-shot mode
1: Continuous operation

Bit 0 **BME**: Burst mode enable

This bit starts the burst mode controller which becomes ready to receive the start trigger. Writing this bit to 0 causes a burst mode early termination.

0: Burst mode disabled
1: Burst mode enabled

33.5.69 HRTIM burst mode trigger register (HRTIM_BMTRGR)

Address offset: 0x724

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OCHPEV	EEVH	EEVG	TDEEVH	TAEEVG	TECMP2	TECMP1	TEREP	TFCMP1	TDCMP2	TFREP	TDREP	TDRST	TFRST	TCCMP1	TCREP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCRST	TBCMP2	TBCMP1	TBREP	TBRST	TACMP2	TACMP1	TAREP	TARST	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTREP	MSTRST	SW
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OCHPEV**: On-chip event

A rising edge on the hrtim_bm_trg input (connected to general purpose timer TRGO output) triggers a burst mode entry (see *Table 325* for details).

Bit 30 **EEVH**: External event H (TIMD filters applied)

The external event H conditioned by TIMD filters is starting the burst mode operation.

Bit 29 **EEVG**: External event G (TIMA filters applied)

The external event G conditioned by TIMA filters is starting the burst mode operation.

Bit 28 **TDEEVH**: Timer D period following external event H

The timer D period following an external event H (conditioned by TIMD filters) is starting the burst mode operation.

Bit 27 **TAEEVG**: Timer A period following external event G

The timer A period following an external event G (conditioned by TIMA filters) is starting the burst mode operation.

Bit 26 **TECMP2**: Timer E compare 2 event

Refer to TACMP1 description.

Bit 25 **TECMP1**: Timer E compare 1 event

Refer to TACMP1 description.

Bit 24 **TEREP**: Timer E repetition

Refer to TAREP description.

Bit 23 **TFCMP1**: Timer F compare 1 event

Refer to TACMP1 description.

Bit 22 **TDCMP2**: Timer D compare 2 event

Refer to TACMP1 description.

Bit 21 **TFREP**: Timer F repetition

Refer to TAREP description.

Bit 20 **TDREP**: Timer D repetition

Refer to TAREP description.

Bit 19 **TDRST**: Timer D reset or roll-over

Refer to TARST description.

Bit 18 **TFRST**: Timer F reset

Refer to TARST description.

Bit 17 **TCCMP1**: Timer C compare 1 event
Refer to TACMP1 description.

Bit 16 **TCREP**: Timer C repetition
Refer to TAREP description.

Bit 15 **TCRST**: Timer C reset or roll-over
Refer to TARST description.

Bit 14 **TBCMP2**: Timer B compare 2 event
Refer to TACMP1 description.

Bit 13 **TBCMP1**: Timer B compare 1 event
Refer to TACMP1 description.

Bit 12 **TBREP**: Timer B repetition
Refer to TAREP description.

Bit 11 **TBRST**: Timer B reset or roll-over
Refer to TARST description.

Bit 10 **TACMP2**: Timer A compare 2 event
Refer to TACMP1 description.

Bit 9 **TACMP1**: Timer A compare 1 event
The timer A compare 1 event is starting the burst mode operation.

Bit 8 **TAREP**: Timer A repetition
The Timer A repetition event is starting the burst mode operation.

Bit 7 **TARST**: Timer A reset or roll-over
The Timer A reset or roll-over event is starting the burst mode operation.

Bit 6 **MSTCMP4**: Master compare 4
Refer to MSTCMP1 description.

Bit 5 **MSTCMP3**: Master compare 3
Refer to MSTCMP1 description.

Bit 4 **MSTCMP2**: Master compare 2
Refer to MSTCMP1 description.

Bit 3 **MSTCMP1**: Master compare 1
The master timer compare 1 event is starting the burst mode operation.

Bit 2 **MSTREP**: Master repetition
The master timer repetition event is starting the burst mode operation.

Bit 1 **MSTRST**: Master reset or roll-over
The master timer reset and roll-over event is starting the burst mode operation.

Bit 0 **SW**: Software start
This bit is set by software and automatically reset by hardware.
When set, It starts the burst mode operation immediately.
This bit is not active if the burst mode is not enabled (BME bit is reset).

33.5.70 HRTIM burst mode compare register (HRTIM_BMCMPR)

Address offset: 0x728

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BMCMP[15:0]**: Burst mode compare value

Defines the number of periods during which the selected timers are in idle state.

This register holds either the content of the preload register or the content of the active register if the preload is disabled.

Note: BMCMP[15:0] cannot be set to 0x0000 when using the f_{HRTIM} clock without a prescaler as the burst mode clock source (BMCLK[3:0] = 1010 and BMPRESC[3:0] = 0000).

33.5.71 HRTIM burst mode period register (HRTIM_BMPER)

Address offset: 0x72C

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BMPER[15:0]**: Burst mode period

Defines the burst mode repetition period.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

Note: The BMPER[15:0] must not be null when the burst mode is enabled.

33.5.72 HRTIM timer external event control register 1 (HRTIM_EECR1)

Address offset: 0x730

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EE5FAST	EE5SNS[1:0]		EE5POL	EE5SRC[1:0]		EE4FAST	EE4SNS[1:0]		EE4POL	EE4SRC[1:0]		EE3FAST	EE3SNS[1]
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE3SNS[0]	EE3POL	EE3SRC[1:0]		EE2FAST	EE2SNS[1:0]		EE2POL	EE2SRC[1:0]		EE1FAST	EE1SNS[1:0]		EE1POL	EE1SRC[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **EE5FAST**: External event 5 fast mode
Refer to EE1FAST description.

Bits 28:27 **EE5SNS[1:0]**: External event 5 sensitivity
Refer to EE1SNS[1:0] description.

Bit 26 **EE5POL**: External event 5 polarity
Refer to EE1POL description.

Bits 25:24 **EE5SRC[1:0]**: External event 5 source
This bitfield is combined with EE5SRC[2] bit. Refer to EE1SRC[1:0] description.

Bit 23 **EE4FAST**: External event 4 fast mode
Refer to EE1FAST description.

Bits 22:21 **EE4SNS[1:0]**: External event 4 sensitivity
Refer to EE1SNS[1:0] description.

Bit 20 **EE4POL**: External event 4 polarity
Refer to EE1POL description.

Bits 19:18 **EE4SRC[1:0]**: External event 4 source
This bitfield is combined with EE4SRC[2] bit. Refer to EE1SRC[1:0] description.

Bit 17 **EE3FAST**: External event 3 fast mode
Refer to EE1FAST description.

Bits 16:15 **EE3SNS[1:0]**: External event 3 sensitivity
Refer to EE1SNS[1:0] description.

Bit 14 **EE3POL**: External event 3 polarity
Refer to EE1POL description.

Bits 13:12 **EE3SRC[1:0]**: External event 3 source
This bitfield is combined with EE3SRC[2] bit. Refer to EE1SRC[1:0] description.

Bit 11 **EE2FAST**: External event 2 fast mode
Refer to EE1FAST description.

Bits 10:9 **EE2SNS[1:0]**: External event 2 sensitivity
Refer to EE1SNS[1:0] description.

Bit 8 **EE2POL**: External event 2 polarity
Refer to EE1POL description.

Bits 7:6 **EE2SRC[1:0]**: External event 2 source
This bitfield is combined with EE2SRC[2] bit. Refer to EE1SRC[1:0] description.

Bit 5 **EE1FAST**: External event 1 fast mode
0: External event 1 is re-synchronized by the HRTIM logic before acting on outputs, which adds a f_{HRTIM} clock-related latency
1: External event 1 is acting asynchronously on outputs (low latency mode)
Note: This bit must not be modified once the counter in which the event is used is enabled (TxCEN bit set).

Bits 4:3 **EE1SNS[1:0]**: External event 1 sensitivity
00: On active level defined by EE1POL bit
01: Rising edge, whatever EE1POL bit value
10: Falling edge, whatever EE1POL bit value
11: Both edges, whatever EE1POL bit value

Bit 2 **EE1POL**: External event 1 polarity
This bit is only significant if EE1SNS[1:0] = 00.
0: External event is active high
1: External event is active low
Note: This parameter cannot be changed once the timer x is enabled. It must be configured prior to setting EE1FAST bit.

Bits 1:0 **EE1SRC[1:0]**: External event 1 source
This bitfield is combined with EE1SRC[2] bit. This bitfield selects the External event 1 source. See *Table 326* for details.
000:hrtim_eev1_1
001:hrtim_eev1_2
010:hrtim_eev1_3
011:hrtim_eev1_4
100:hrtim_eev1_5
001:hrtim_eev1_6
010:hrtim_eev1_7
011:hrtim_eev1_8
Note: This parameter cannot be changed once the timer x is enabled. It must be configured prior to setting EE1FAST bit.

33.5.73 HRTIM timer external event control register 2 (HRTIM_EECR2)

Address offset: 0x734

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EE10SNS[1:0]		EE10POL	EE10SRC[1:0]		Res.	EE9SNS[1:0]		EE9POL	EE9SRC[1:0]		Res.	EE8SNS[1]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE8SNS[0]	EE8POL	EE8SRC[1:0]		Res.	EE7SNS[1:0]		EE7POL	EE7SRC[1:0]		Res.	EE6SNS[1:0]		EE6POL	EE6SRC[1:0]	
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:27 **EE10SNS[1:0]**: External event 10 sensitivity
Refer to EE1SNS[1:0] description.

Bit 26 **EE10POL**: External event 10 polarity
Refer to EE1POL description.

Bits 25:24 **EE10SRC[1:0]**: External event 10 source
This bitfield is combined with EE10SRC[2] bit. Refer to EE1SRC[1:0] description.

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **EE9SNS[1:0]**: External event 9 sensitivity
Refer to EE1SNS[1:0] description.

Bit 20 **EE9POL**: External event 9 polarity
Refer to EE1POL description.

Bits 19:18 **EE9SRC[1:0]**: External event 9 source
This bitfield is combined with EE9SRC[2] bit. Refer to EE1SRC[1:0] description.

Bit 17 Reserved, must be kept at reset value.

Bits 16:15 **EE8SNS[1:0]**: External event 8 sensitivity
Refer to EE1SNS[1:0] description.

Bit 14 **EE8POL**: External event 8 polarity
Refer to EE1POL description.

Bits 13:12 **EE8SRC[1:0]**: External event 8 source
This bitfield is combined with EE8SRC[2] bit. Refer to EE1SRC[1:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:9 **EE7SNS[1:0]**: External event 7 sensitivity
Refer to EE1SNS[1:0] description.

Bit 8 **EE7POL**: External event 7 polarity
Refer to EE1POL description.

Bits 7:6 **EE7SRC[1:0]**: External event 7 source
This bitfield is combined with EE7SRC[2] bit. Refer to EE1SRC[1:0] description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **EE6SNS[1:0]**: External event 6 sensitivity
Refer to EE1SNS[1:0] description.

Bit 2 **EE6POL**: External event 6 polarity
Refer to EE1POL description.

Bits 1:0 **EE6SRC[1:0]**: External event 6 source
This bitfield is combined with EE6SRC[2] bit. Refer to EE1SRC[1:0] description.

33.5.74 HRTIM timer external event control register 3 (HRTIM_EECR3)

Address offset: 0x738

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EEVSD[1:0]		Res.	Res.	EE10F[3:0]				Res.	Res.	EE9F[3:0]				Res.	Res.
rw	rw			rw	rw	rw	rw			rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE8F[3:0]				Res.	Res.	EE7F[3:0]				Res.	Res.	EE6F[3:0]			
rw	rw	rw	rw			rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:30 **EEVSD[1:0]**: External event sampling clock division
This bitfield indicates the division ratio between the timer clock frequency (f_{HRTIM}) and the external event signal sampling clock (f_{EEVS}) used by the digital filters.
00: $f_{EEVS}=f_{HRTIM}$
01: $f_{EEVS}=f_{HRTIM}$ / 2
10: $f_{EEVS}=f_{HRTIM}$ / 4
11: $f_{EEVS}=f_{HRTIM}$ / 8

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:24 **EE10F[3:0]**: External event 10 filter
Refer to EE6F[3:0] description.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:18 **EE9F[3:0]**: External event 9 filter
Refer to EE6F[3:0] description.

Bits 17:16 Reserved, must be kept at reset value.

Bits 15:12 **EE8F[3:0]**: External event 8 filter
Refer to EE6F[3:0] description.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:6 **EE7F[3:0]**: External event 7 filter
Refer to EE6F[3:0] description.

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **EE6F[3:0]**: External event 6 filter
This bitfield defines the frequency used to sample external event 6 input and the length of the digital filter applied to EEV6. The digital filter is made of a counter in which N valid samples are needed to validate a transition on the output.
0000: Filter disabled
0001: $f_{SAMPLING} = f_{HRTIM}$, N=2
0010: $f_{SAMPLING} = f_{HRTIM}$, N=4
0011: $f_{SAMPLING} = f_{HRTIM}$, N=8
0100: $f_{SAMPLING} = f_{EEVS}/2$, N=6
0101: $f_{SAMPLING} = f_{EEVS}/2$, N=8
0110: $f_{SAMPLING} = f_{EEVS}/4$, N=6
0111: $f_{SAMPLING} = f_{EEVS}/4$, N=8
1000: $f_{SAMPLING} = f_{EEVS}/8$, N=6
1001: $f_{SAMPLING} = f_{EEVS}/8$, N=8
1010: $f_{SAMPLING} = f_{EEVS}/16$, N=5
1011: $f_{SAMPLING} = f_{EEVS}/16$, N=6
1100: $f_{SAMPLING} = f_{EEVS}/16$, N=8
1101: $f_{SAMPLING} = f_{EEVS}/32$, N=5
1110: $f_{SAMPLING} = f_{EEVS}/32$, N=6
1111: $f_{SAMPLING} = f_{EEVS}/32$, N=8

33.5.75 HRTIM ADC trigger 1 register (HRTIM_ADC1R)

Address offset: 0x73C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC1TEPER	ADC1TEC4	ADC1TEC3	ADC1TFRST	ADC1TDPER	ADC1TDC4	ADC1TDC3	ADC1TFPER	ADC1TCPER	ADC1TCC4	ADC1TCC3	ADC1TFC4	ADC1TBRST	ADC1TBPER	ADC1TBC4	ADC1TBC3
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC1TFC3	ADC1TARST	ADC1TAPER	ADC1TAC4	ADC1TAC3	ADC1TFC2	ADC1EEV5	ADC1EEV4	ADC1EEV3	ADC1EEV2	ADC1EEV1	ADC1MPER	ADC1MC4	ADC1MC3	ADC1MC2	ADC1MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADC1TEPER**: ADC trigger 1 on timer E period
Refer to ADC1TAPER description.

Bit 30 **ADC1TEC4**: ADC trigger 1 on timer E compare 4
Refer to ADC1TFC2 description.

Bit 29 **ADC1TEC3**: ADC trigger 1 on timer E compare 3
Refer to ADC1TFC2 description.

Bit 28 **ADC1TFRST**: ADC trigger 1 on timer F reset and counter roll-over(1)
Refer to ADC1TARST description.

Bit 27 **ADC1TDPER**: ADC trigger 1 on timer D period
Refer to ADC1TAPER description.

Bit 26 **ADC1TDC4**: ADC trigger 1 on timer D compare 4
Refer to ADC1TFC2 description.

Bit 25 **ADC1TDC3**: ADC trigger 1 on timer D compare 3
Refer to ADC1TFC2 description.

Bit 24 **ADC1TFPER**: ADC trigger 1 on timer F period
Refer to ADC1TAPER description.

Bit 23 **ADC1TCPER**: ADC trigger 1 on timer C period
Refer to ADC1TAPER description.

Bit 22 **ADC1TCC4**: ADC trigger 1 on timer C compare 4
Refer to ADC1TFC2 description.

Bit 21 **ADC1TCC3**: ADC trigger 1 on timer C compare 3
Refer to ADC1TFC2 description.

Bit 20 **ADC1TFC4**: ADC trigger 1 on timer F compare 4
Refer to ADC1TAC2 description.

Bit 19 **ADC1TBRST**: ADC trigger 1 on timer B reset and counter roll-over(1)
This bit enables the generation of an ADC trigger upon timer B reset and roll-over event, on ADC trigger 1 output.

Bit 18 **ADC1TBPER**: ADC trigger 1 on timer B period
Refer to ADC1TAPER description.

Bit 17 **ADC1TBC4**: ADC trigger 1 on timer B compare 4
Refer to ADC1TFC2 description.

Bit 16 **ADC1TBC3**: ADC trigger 1 on timer B compare 3
Refer to ADC1TFC2 description.

Bit 15 **ADC1TFC3**: ADC trigger 1 on timer F compare 3
Refer to ADC1TFC2 description.

Bit 14 **ADC1TARST**: ADC trigger 1 on timer A reset and counter roll-over(1)
This bit enables the generation of an ADC trigger upon timer A reset and roll-over event, on ADC trigger 1 output.

Bit 13 **ADC1TAPER**: ADC trigger 1 on timer A period
This bit enables the generation of an ADC trigger upon timer A period event, on ADC trigger 1 output.

Bit 12 **ADC1TAC4**: ADC trigger 1 on timer A compare 4
Refer to ADC1TFC2 description.

Bit 11 **ADC1TAC3**: ADC trigger 1 on timer A compare 3
Refer to ADC1TFC2 description.

Bit 10 **ADC1TFC2**: ADC trigger 1 on timer F compare 2
This bit enables the generation of an ADC trigger upon timer F compare 2 event, on ADC trigger 1 output.

Bit 9 **ADC1EEV5**: ADC trigger 1 on external event 5(1)
Refer to ADC1EEV1 description.

Bit 8 **ADC1EEV4**: ADC trigger 1 on external event 4(1)
Refer to ADC1EEV1 description.

Bit 7 **ADC1EEV3**: ADC trigger 1 on external event 3(1)
Refer to ADC1EEV1 description.

Bit 6 **ADC1EEV2**: ADC trigger 1 on external event 2(1)
Refer to ADC1EEV1 description.

Bit 5 **ADC1EEV1**: ADC trigger 1 on external event 1(1)
This bit enables the generation of an ADC trigger upon external event 1, on ADC trigger 1 output.

Bit 4 **ADC1MPER**: ADC trigger 1 on master period
This bit enables the generation of an ADC trigger upon master timer period event, on ADC trigger 1 output.

Bit 3 **ADC1MC4**: ADC trigger 1 on master compare 4
Refer to ADC1MC1 description.

Bit 2 **ADC1MC3**: ADC trigger 1 on master compare 3
Refer to ADC1MC1 description.

Bit 1 **ADC1MC2**: ADC trigger 1 on master compare 2
Refer to ADC1MC1 description.

Bit 0 **ADC1MC1**: ADC trigger 1 on master compare 1
This bit enables the generation of an ADC trigger upon master compare 1 event, on ADC trigger 1 output.

1. These triggers are differing from HRTIM_ADC2R/HRTIM_ADC4R to HRTIM_ADC1R/HRTIM_ADC3R.

33.5.76 HRTIM ADC trigger 2 register (HRTIM_ADC2R)

Address offset: 0x740

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC2TERST	ADC2TEC4	ADC2TEC3	ADC2TEC2	ADC2TDRST	ADC2TDPER	ADC2TDC4	ADC2TFPER	ADC2TDC2	ADC2TCRST	ADC2TCPER	ADC2TCC4	ADC2TFC4	ADC2TCC2	ADC2TBPER	ADC2TBC4
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC2TFC3	ADC2TBC2	ADC2TAPER	ADC2TAC4	ADC2TFC2	ADC2TAC2	ADC2EEV10	ADC2EEV9	ADC2EEV8	ADC2EEV7	ADC2EEV6	ADC2MPER	ADC2MC4	ADC2MC3	ADC2MC2	ADC2MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADC2TERST**: ADC trigger 2 on timer E reset and counter roll-over(1)
Refer to ADC2TCRST description.

Bit 30 **ADC2TEC4**: ADC trigger 2 on timer E compare 4
Refer to ADC2TAC2 description.

Bit 29 **ADC2TEC3**: ADC trigger 2 on timer E compare 3
Refer to ADC2TAC2 description.

Bit 28 **ADC2TEC2**: ADC trigger 2 on timer E compare 2
Refer to ADC2TAC2 description.

Bit 27 **ADC2TDRST**: ADC trigger 2 on timer D reset and counter roll-over(1)
Refer to ADC2TCRST description.

Bit 26 **ADC2TDPER**: ADC trigger 2 on timer D period
Refer to ADC2TAPER description.

Bit 25 **ADC2TDC4**: ADC trigger 2 on timer D compare 4
Refer to ADC2TAC2 description.

Bit 24 **ADC2TFPER**: ADC trigger 2 on timer F period
Refer to ADC2TAPER description.

Bit 23 **ADC2TDC2**: ADC trigger 2 on timer D compare 2
Refer to ADC2TAC2 description.

Bit 22 **ADC2TCRST**: ADC trigger 2 on timer C reset and counter roll-over(1)
This bit enables the generation of an ADC trigger upon timer C reset and roll-over event, on ADC trigger 1 output.

Bit 21 **ADC2TCPER**: ADC trigger 2 on timer C period
Refer to ADC2TAPER description.

Bit 20 **ADC2TCC4**: ADC trigger 2 on timer C compare 4
Refer to ADC2TAC2 description.

Bit 19 **ADC2TFC4**: ADC trigger 2 on timer F compare 4
Refer to ADC2TAC2 description.

Bit 18 **ADC2TCC2**: ADC trigger 2 on timer C compare 2
Refer to ADC2TAC2 description.

Bit 17 **ADC2TBPER**: ADC trigger 2 on timer B period
Refer to ADC2TAPER description.

Bit 16 **ADC2TBC4**: ADC trigger 2 on timer B compare 4
Refer to ADC2TAC2 description.

Bit 15 **ADC2TFC3**: ADC trigger 2 on timer F compare 3
Refer to ADC2TAC2 description.

Bit 14 **ADC2TBC2**: ADC trigger 2 on timer B compare 2
Refer to ADC2TAC2 description.

Bit 13 **ADC2TAPER**: ADC trigger 2 on timer A period
This bit enables the generation of an ADC trigger upon timer A event, on ADC trigger 2 output.

Bit 12 **ADC2TAC4**: ADC trigger 2 on timer A compare 4
Refer to ADC2TAC2 description.

Bit 11 **ADC2TFC2**: ADC trigger 2 on timer F compare 2
Refer to ADC2TAC2 description.

Bit 10 **ADC2TAC2**: ADC trigger 2 on timer A compare 2
This bit enables the generation of an ADC trigger upon timer A compare 2, on ADC trigger 2 output.

Bit 9 **ADC2EEV10**: ADC trigger 2 on external event 10(1)
Refer to ADC2EEV6 description.

Bit 8 **ADC2EEV9**: ADC trigger 2 on external event 9(1)
Refer to ADC2EEV6 description.

Bit 7 **ADC2EEV8**: ADC trigger 2 on external event 8(1)
Refer to ADC2EEV6 description.

Bit 6 **ADC2EEV7**: ADC trigger 2 on external event 7(1)
Refer to ADC2EEV6 description.

Bit 5 **ADC2EEV6**: ADC trigger 2 on external event 6(1)
This bit enables the generation of an ADC trigger upon external event 6, on ADC trigger 2 output.

Bit 4 **ADC2MPER**: ADC trigger 2 on master period
This bit enables the generation of an ADC trigger upon master period event, on ADC trigger 2 output.

Bit 3 **ADC2MC4**: ADC trigger 2 on master compare 4
Refer to ADC2MC1 description.

Bit 2 **ADC2MC3**: ADC trigger 2 on master compare 3
Refer to ADC2MC1 description.

Bit 1 **ADC2MC2**: ADC trigger 2 on master compare 2
Refer to ADC2MC1 description.

Bit 0 **ADC2MC1**: ADC trigger 2 on master compare 1
This bit enables the generation of an ADC trigger upon master compare 1 event, on ADC trigger 2 output.

1. These triggers are differing from HRTIM_ADC1R/HRTIM_ADC3R to HRTIM_ADC2R/HRTIM_ADC4R.

33.5.77 HRTIM ADC trigger 3 register (HRTIM_ADC3R)

Address offset: 0x744

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC3TEPER	ADC3TEC4	ADC3TEC3	ADC3TFRST	ADC3TDPER	ADC3TDC4	ADC3TDC3	ADC3TFPER	ADC3TCPER	ADC3TCC4	ADC3TCC3	ADC3TFC4	ADC3TBRST	ADC3TBPER	ADC3TBC4	ADC3TBC3
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3TFC3	ADC3TARST	ADC3TAPER	ADC3TAC4	ADC3TAC3	ADC3TFC2	ADC3EEV5	ADC3EEV4	ADC3EEV3	ADC3EEV2	ADC3EEV1	ADC3MPER	ADC3MC4	ADC3MC3	ADC3MC2	ADC3MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADC3TEPER**: ADC trigger 3 on timer E period
Refer to ADC3TAPER description.

Bit 30 **ADC3TEC4**: ADC trigger 3 on timer E compare 4
Refer to ADC3TFC2 description.

Bit 29 **ADC3TEC3**: ADC trigger 3 on timer E compare 3
Refer to ADC3TFC2 description.

Bit 28 **ADC3TFRST**: ADC trigger 3 on timer F reset and counter roll-over(1)
Refer to ADC3TARST description.

Bit 27 **ADC3TDPER**: ADC trigger 3 on timer D period
Refer to ADC3TAPER description.

Bit 26 **ADC3TDC4**: ADC trigger 3 on timer D compare 4
Refer to ADC3TFC2 description.

Bit 25 **ADC3TDC3**: ADC trigger 3 on timer D compare 3
Refer to ADC3TFC2 description.

Bit 24 **ADC3TFPER**: ADC trigger 3 on timer F period
Refer to ADC3TAPER description.

Bit 23 **ADC3TCPER**: ADC trigger 3 on timer C period
Refer to ADC3TAPER description.

Bit 22 **ADC3TCC4**: ADC trigger 3 on timer C compare 4
Refer to ADC3TFC2 description.

Bit 21 **ADC3TCC3**: ADC trigger 3 on timer C compare 3
Refer to ADC3TFC2 description.

Bit 20 **ADC3TFC4**: ADC trigger 3 on timer F compare 4
Refer to ADC3TAC2 description.

Bit 19 **ADC3TBRST**: ADC trigger 3 on timer B reset and counter roll-over(1)
This bit enables the generation of an ADC trigger upon timer B reset and roll-over event, on ADC trigger 1 output.

Bit 18 **ADC3TBPER**: ADC trigger 3 on timer B period
Refer to ADC3TAPER description.

Bit 17 **ADC3TBC4**: ADC trigger 3 on timer B compare 4
Refer to ADC3TFC2 description.

Bit 16 **ADC3TBC3**: ADC trigger 3 on timer B compare 3
Refer to ADC3TFC2 description.

Bit 15 **ADC3TFC3**: ADC trigger 3 on timer F compare 3
Refer to ADC3TFC2 description.

Bit 14 **ADC3TARST**: ADC trigger 3 on timer A reset and counter roll-over(1)
This bit enables the generation of an ADC trigger upon timer A reset and roll-over event, on ADC trigger 1 output.

Bit 13 **ADC3TAPER**: ADC trigger 3 on timer A period
This bit enables the generation of an ADC trigger upon timer A period event, on ADC trigger 1 output.

Bit 12 **ADC3TAC4**: ADC trigger 3 on timer A compare 4
Refer to ADC3TFC2 description.

Bit 11 **ADC3TAC3**: ADC trigger 3 on timer A compare 3
Refer to ADC3TFC2 description.

Bit 10 **ADC3TFC2**: ADC trigger 3 on timer F compare 2
This bit enables the generation of an ADC trigger upon timer F compare 2 event, on ADC trigger 3 output.

Bit 9 **ADC3EEV5**: ADC trigger 3 on external event 5(1)
Refer to ADC3EEV1 description.

Bit 8 **ADC3EEV4**: ADC trigger 3 on external event 4(1)
Refer to ADC3EEV1 description.

Bit 7 **ADC3EEV3**: ADC trigger 3 on external event 3(1)
Refer to ADC3EEV1 description.

Bit 6 **ADC3EEV2**: ADC trigger 3 on external event 2(1)
Refer to ADC3EEV1 description.

Bit 5 **ADC3EEV1**: ADC trigger 3 on external event 1(1)
This bit enables the generation of an ADC trigger upon external event 1, on ADC trigger 3 output.

Bit 4 **ADC3MPER**: ADC trigger 3 on master period
This bit enables the generation of an ADC trigger upon master timer period event, on ADC trigger 3 output.

Bit 3 **ADC3MC4**: ADC trigger 3 on master compare 4
Refer to ADC3MC1 description.

Bit 2 **ADC3MC3**: ADC trigger 3 on master compare 3
Refer to ADC3MC1 description.

Bit 1 **ADC3MC2**: ADC trigger 3 on master compare 2
Refer to ADC3MC1 description.

Bit 0 **ADC3MC1**: ADC trigger 3 on master compare 1
This bit enables the generation of an ADC trigger upon master compare 1 event, on ADC trigger 3 output.

1. These triggers are differing from HRTIM_ADC2R/HRTIM_ADC4R to HRTIM_ADC1R/HRTIM_ADC3R.

33.5.78 HRTIM ADC trigger 4 register (HRTIM_ADC4R)

Address offset: 0x748

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC4TERST	ADC4TEC4	ADC4TEC3	ADC4TEC2	ADC4TDRST	ADC4TDPER	ADC4TDC4	ADC4TFPER	ADC4TDC2	ADC4TCRST	ADC4TCPER	ADC4TCC4	ADC4TFC4	ADC4TCC2	ADC4TBPER	ADC4TBC4
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC4TFC3	ADC4TBC2	ADC4TAPER	ADC4TAC4	ADC4TFC2	ADC4TAC2	ADC4EEV10	ADC4EEV9	ADC4EEV8	ADC4EEV7	ADC4EEV6	ADC4MPER	ADC4MC4	ADC4MC3	ADC4MC2	ADC4MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADC4TERST**: ADC trigger 4 on timer E reset and counter roll-over(1)
Refer to ADC4TCRST description.

Bit 30 **ADC4TEC4**: ADC trigger 4 on timer E compare 4
Refer to ADC4TAC2 description.

Bit 29 **ADC4TEC3**: ADC trigger 4 on timer E compare 3
Refer to ADC4TAC2 description.

Bit 28 **ADC4TEC2**: ADC trigger 4 on timer E compare 2
Refer to ADC4TAC2 description.

Bit 27 **ADC4TDRST**: ADC trigger 4 on timer D reset and counter roll-over(1)
Refer to ADC4TCRST description.

Bit 26 **ADC4TDPER**: ADC trigger 4 on timer D period
Refer to ADC4TAPER description.

Bit 25 **ADC4TDC4**: ADC trigger 4 on timer D compare 4
Refer to ADC4TAC2 description.

Bit 24 **ADC4TFPER**: ADC trigger 4 on timer F period
Refer to ADC4TAPER description.

Bit 23 **ADC4TDC2**: ADC trigger 2 on timer D compare 2
Refer to ADC4TAC2 description.

Bit 22 **ADC4TCRST**: ADC trigger 4 on timer C reset and counter roll-over(1)
This bit enables the generation of an ADC trigger upon timer C reset and roll-over event, on ADC trigger 1 output.

Bit 21 **ADC4TCPER**: ADC trigger 4 on timer C period
Refer to ADC4TAPER description.

Bit 20 **ADC4TCC4**: ADC trigger 4 on timer C compare 4
Refer to ADC4TAC2 description.

Bit 19 **ADC4TFC4**: ADC trigger 4 on timer F compare 4
Refer to ADC4TAC2 description.

Bit 18 **ADC4TCC2**: ADC trigger 4 on timer C compare 2
Refer to ADC4TAC2 description.

Bit 17 **ADC4TBPER**: ADC trigger 4 on timer B period
Refer to ADC4TAPER description.

Bit 16 **ADC4TBC4**: ADC trigger 4 on timer B compare 4
Refer to ADC4TAC2 description.

Bit 15 **ADC4TFC3**: ADC trigger 4 on timer F compare 3
Refer to ADC4TAC2 description.

Bit 14 **ADC4TBC2**: ADC trigger 4 on timer B compare 2
Refer to ADC4TAC2 description.

Bit 13 **ADC4TAPER**: ADC trigger 4 on timer A period
This bit enables the generation of an ADC trigger upon timer A event, on ADC trigger 2 output.

Bit 12 **ADC4TAC4**: ADC trigger 4 on timer A compare 4
Refer to ADC4TAC2 description.

Bit 11 **ADC4TFC2**: ADC trigger 4 on timer F compare 2
Refer to ADC4TAC2 description.

Bit 10 **ADC4TAC2**: ADC trigger 4 on timer A compare 2
This bit enables the generation of an ADC trigger upon timer A compare 2, on ADC trigger 2 output.

Bit 9 **ADC4EEV10**: ADC trigger 4 on external event 10(1)
Refer to ADC4EEV6 description.

Bit 8 **ADC4EEV9**: ADC trigger 4 on external event 9(1)
Refer to ADC4EEV6 description.

Bit 7 **ADC4EEV8**: ADC trigger 4 on external event 8(1)
Refer to ADC4EEV6 description.

Bit 6 **ADC4EEV7**: ADC trigger 4 on external event 7(1)
Refer to ADC4EEV6 description.

Bit 5 **ADC4EEV6**: ADC trigger 4 on external event 6(1)
This bit enables the generation of an ADC trigger upon external event 6, on ADC trigger 2 output.

Bit 4 **ADC4MPER**: ADC trigger 4 on master period
This bit enables the generation of an ADC trigger upon master period event, on ADC trigger 2 output.

Bit 3 **ADC4MC4**: ADC trigger 4 on master compare 4
Refer to ADC4MC1 description.

Bit 2 **ADC4MC3**: ADC trigger 4 on master compare 3
Refer to ADC4MC1 description.

Bit 1 **ADC4MC2**: ADC trigger 4 on master compare 2
Refer to ADC4MC1 description.

Bit 0 **ADC4MC1**: ADC trigger 4 on master compare 1
This bit enables the generation of an ADC trigger upon master compare 1 event, on ADC trigger 2 output.

1. These triggers are differing from HRTIM_ADC1R/HRTIM_ADC3R to HRTIM_ADC2R/HRTIM_ADC4R.

33.5.79 HRTIM fault input register 1 (HRTIM_FLTINR1)

Address offset: 0x750

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLT4LCK	FLT4F[3:0]				FLT4SRC[0]	FLT4P	FLT4E	FLT3LCK	FLT3F[3:0]				FLT3SRC[0]	FLT3P	FLT3E
rwo	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT2LCK	FLT2F[3:0]				FLT2SRC[0]	FLT2P	FLT2E	FLT1LCK	FLT1F[3:0]				FLT1SRC[0]	FLT1P	FLT1E
rwo	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLT4LCK**: Fault 4 lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register.

Bits 30:27 **FLT4F[3:0]**: Fault 4 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register.

Bit 26 FLT4SRC[0]: Fault 4 source bit 0
Refer to FLT5SRC[0] description in HRTIM_FLTINR2 register.

Bit 25 **FLT4P**: Fault 4 polarity
Refer to FLT5P description in HRTIM_FLTINR2 register.

Bit 24 **FLT4E**: Fault 4 enable
Refer to FLT5E description in HRTIM_FLTINR2 register.

Bit 23 **FLT3LCK**: Fault 3 lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register.

Bits 22:19 **FLT3F[3:0]**: Fault 3 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register.

Bit 18 FLT3SRC[0]: Fault 3 source bit 0
Refer to FLT5SRC[0] description in HRTIM_FLTINR2 register.

Bit 17 **FLT3P**: Fault 3 polarity
Refer to FLT5P description in HRTIM_FLTINR2 register.

Bit 16 **FLT3E**: Fault 3 enable
Refer to FLT5E description in HRTIM_FLTINR2 register.

Bit 15 **FLT2LCK**: Fault 2 lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register.

Bits 14:11 **FLT2F[3:0]**: Fault 2 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register.

Bit 10 FLT2SRC[0]: Fault 2 source bit 0
Refer to FLT5SRC[0] description in HRTIM_FLTINR2 register.

Bit 9 **FLT2P**: Fault 2 polarity
Refer to FLT2P description in HRTIM_FLTINR2 register.

Bit 8 **FLT2E**: Fault 2 enable
Refer to FLT5E description in HRTIM_FLTINR2 register.

Bit 7 **FLT1LCK**: Fault 1 lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register.

Bits 6:3 **FLT1F[3:0]**: Fault 1 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register.

Bit 2 **FLT1SRC[0]**: Fault 1 source bit 0
Refer to FLT5SRC[0] description in HRTIM_FLTINR2 register.

Bit 1 **FLT1P**: Fault 1 polarity
Refer to FLT5P description in HRTIM_FLTINR2 register.

Bit 0 **FLT1E**: Fault 1 enable
Refer to FLT5E description in HRTIM_FLTINR2 register.

33.5.80 HRTIM fault input register 2 (HRTIM_FLTINR2)

Address offset: 0x754

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FLTSD[1:0]		FLT8SRC[1]	FLT7SRC[1]	FLT6SRC[1]	FLT5SRC[1]	FLT4SRC[1]	FLT3SRC[1]	FLT2SRC[1]	FLT1SRC[1]
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT6LCK	FLT6F[3:0]				FLT6SRC[0]	FLT6P	FLT6E	FLT5LCK	FLT5F[3:0]				FLT5SRC[0]	FLT5P	FLT5E
rwo	rw	rw	rw	rw	rw	rw	rw	rwo	rw	rw	rw	rw	rw	rw	rw

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

Bits 25:24 **FLTSD[1:0]**: Fault sampling clock division

This bitfield indicates the division ratio between the timer clock frequency (f_{HRTIM}) and the fault signal sampling clock (f_{FLTS}) used by the digital filters.

00: $f_{FLTS}=f_{HRTIM}$

01: $f_{FLTS}=f_{HRTIM}$ / 2

10: $f_{FLTS}=f_{HRTIM}$ / 4

11: $f_{FLTS}=f_{HRTIM}$ / 8

Note: This bitfield must be written prior to any of the FLTxE enable bits.

Bit 23 **FLT8SRC[1]**: Fault 8 source bit 1

Refer to FLT5SRC[0] description.

Bit 22 **FLT7SRC[1]**: Fault 7 source bit 1

Refer to FLT5SRC[0] description.

Bit 21 **FLT6SRC[1]**: Fault 6 source bit 1

Refer to FLT5SRC[0] description.

Bit 20 **FLT5SRC[1]**: Fault 5 source bit 1

Refer to FLT5SRC[0] description.

Bit 19 **FLT4SRC[1]**: Fault 4 source bit 1

Refer to FLT5SRC[0] description.

Bit 18 **FLT3SRC[1]**: Fault 3 source bit 1

Refer to FLT5SRC[0] description.

Bit 17 **FLT2SRC[1]**: Fault 2 source bit 1

Refer to FLT5SRC[0] description.

Bit 16 **FLT1SRC[1]**: Fault 1 source bit 1

Refer to FLT5SRC[0] description.

Bit 15 **FLT6LCK**: Fault 6 lock

Refer to FLT5LCK description.

Bits 14:11 **FLT6F[3:0]**: Fault 6 filter

Refer to FLT5F[3:0] description.

Bit 10 **FLT6SRC[0]**: Fault 6 source bit 0

Refer to FLT5SRC[0] description.

Bit 9 **FLT6P**: Fault 6 polarity
Refer to FLT5P description.

Bit 8 **FLT6E**: Fault 6 enable
Refer to FLT5E description.

Bit 7 **FLT5LCK**: Fault 5 lock
The FLT5LCK bit modifies the write attributes of the fault programming bit, so that they are protected against spurious write accesses.
This bit is write-once. Once it has been set, it cannot be modified till the next system reset.
0: FLT5E, FLT5P, FLT5SRC[1:0], FLT5F[3:0] and FLT5BLK bits are read/write.
1: FLT5E, FLT5P, FLT5SRC[1:0], FLT5F[3:0] and FLT5BLK bits cannot be written (read-only mode)

Bits 6:3 **FLT5F[3:0]**: Fault 5 filter
This bitfield defines the frequency used to sample FLT5 input and the length of the digital filter applied to FLT5. The digital filter is made of an event counter in which N events are needed to validate a transition on the output.
0000: No filter, FLT5 acts asynchronously
0001: $f_{SAMPLING} = f_{HRTIM}$, N = 2
0010: $f_{SAMPLING} = f_{HRTIM}$, N = 4
0011: $f_{SAMPLING} = f_{HRTIM}$, N = 8
0100: $f_{SAMPLING} = f_{FLTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{FLTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{FLTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{FLTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{FLTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{FLTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{FLTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{FLTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{FLTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{FLTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{FLTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{FLTS}/32$, N = 8
Note: This bitfield can be written only when FLT5E enable bit is reset.
This bitfield cannot be modified when FLT5LOCK has been programmed.

Bit 2 **FLT5SRC[0]**: Fault 5 source bit 0
The FTL5SRC[1:0] bitfield selects the FAULT5 input source (refer to *Table 325* for connection details).
00: Fault 5 input is HRTIM_FLT5 input pin
01: Fault 5 input is connected to a COMPx output
10: Fault 5 input is EEV5_muxout input pin
11: hrtim_ext_trg5
Note: This bitfield can be written only when FLT5E enable bit is reset.

Bit 1 **FLT5P**: Fault 5 polarity
This bit selects the FAULT5 input polarity.
0: Fault 5 input is active low
1: Fault 5 input is active high
Note: This bitfield can be written only when FLT5E enable bit is reset.

Bit 0 **FLT5E**: Fault 5 enable
This bit enables the global FAULT5 input circuitry.
0: Fault 5 input disabled
1: Fault 5 input enabled

33.5.81 HRTIM burst DMA master timer update register (HRTIM_BDMUPR)

Address offset: 0x758

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MCMP4	MCMP3	MCMP2	MCMP1	MREP	MPER	MCNT	MDIER	MICR	MCR
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 9 **MCMP4**: MCMP4R register update enable
Refer to MCR description.

Bit 8 **MCMP3**: MCMP3R register update enable
Refer to MCR description.

Bit 7 **MCMP2**: MCMP2R register update enable
Refer to MCR description.

Bit 6 **MCMP1**: MCMP1R register update enable
Refer to MCR description.

Bit 5 **MREP**: MREP register update enable
Refer to MCR description.

Bit 4 **MPER**: MPER register update enable
Refer to MCR description.

Bit 3 **MCNT**: MCNTR register update enable
Refer to MCR description.

Bit 2 **MDIER**: MDIER register update enable
Refer to MCR description.

Bit 1 **MICR**: MICR register update enable
Refer to MCR description.

Bit 0 **MCR**: MCR register update enable
This bit defines if the master timer MCR register is part of the list of registers to be updated by the burst DMA.
0: MCR register is not updated by burst DMA accesses
1: MCR register is updated by burst DMA accesses

33.5.82 HRTIM burst DMA timer x update register (HRTIM_BDTxUPR) (x = A to F)

Address offset: Block A: 0x75C

Address offset: Block B: 0x760

Address offset: Block C: 0x764

Address offset: Block D: 0x768

Address offset: Block E: 0x76C

Address offset: Block F: 0x774

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	TIMxCODER	TIMxCOFER	TIMxCOTRGR	TIMxEESELR	TIMxEEFR3	TIMxCR2	TIMxFLTR	TIMxOUTR	TIMxCHPR	TIMxRSTR	TIMxEEFR2
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMxEEFR1	TIMxRST2R	TIMxSET2R	TIMxRST1R	TIMxSET1R	TIMxDTR	TIMxCMP4	TIMxCMP3	TIMxCMP2	TIMxCMP1	TIMxREP	TIMxPER	TIMxCNT	TIMxDIER	TIMxICR	TIMxCR
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 26 **TIMxCODER**: HRTIM_CODExR register update enable timer x
Refer to TIMxCR description.

Bit 25 **TIMxCOFER**: HRTIM_COFExR register update enable timer x
Refer to TIMxCR description.

Bit 24 **TIMxCOTRGR**: HRTIM_COTRGxR register update enable timer x
Refer to TIMxCR description.

Bit 23 **TIMxEESELR**: HRTIM_EESELxR register update enable timer x
Refer to TIMxCR description.

Bit 22 **TIMxEEFR3**: HRTIM_EEFxR3 register update enable timer x
Refer to TIMxCR description.

Bit 21 **TIMxCR2**: HRTIM_TIMxCR2 register update enable timer x
Refer to TIMxCR description.

Bit 20 **TIMxFLTR**: HRTIM_FLTxR register update enable timer x
Refer to TIMxCR description.

Bit 19 **TIMxOUTR**: HRTIM_OUTxR register update enable timer x
Refer to TIMxCR description.

Bit 18 **TIMxCHPR**: HRTIM_CHPxR register update enable timer x
Refer to TIMxCR description.

Bit 17 **TIMxRSTR**: HRTIM_RSTxR register update enable timer x
Refer to TIMxCR description.

Bit 16 **TIMxEEFR2**: HRTIM_EEFxR2 register update enable timer x
Refer to TIMxCR description.

Bit 15 **TIMxEEFR1**: HRTIM_EEFxR1 register update enable timer x
Refer to TIMxCR description.

Bit 14 **TIMxRST2R**: HRTIM_RST2xR register update enable timer x
Refer to TIMxCR description.

Bit 13 **TIMxSET2R**: HRTIM_SET2xR register update enable timer x
Refer to TIMxCR description.

Bit 12 **TIMxRST1R**: HRTIM_RST1xR register update enable timer x
Refer to TIMxCR description.

Bit 11 **TIMxSET1R**: HRTIM_SET1xR register update enable timer x
Refer to TIMxCR description.

Bit 10 **TIMxDTR**: HRTIM_DTxR register update enable timer x
Refer to TIMxCR description.

Bit 9 **TIMxCMP4**: HRTIM_CMP4xR register update enable timer x
Refer to TIMxCR description.

Bit 8 **TIMxCMP3**: HRTIM_CMP3xR register update enable timer x
Refer to TIMxCR description.

Bit 7 **TIMxCMP2**: HRTIM_CMP2xR register update enable timer x
Refer to TIMxCR description.

Bit 6 **TIMxCMP1**: HRTIM_CMP1xR register update enable timer x
Refer to TIMxCR description.

Bit 5 **TIMxREP**: HRTIM_REPxR register update enable timer x
Refer to TIMxCR description.

Bit 4 **TIMxPER**: HRTIM_PERxR register update enable timer x
Refer to TIMxCR description.

Bit 3 **TIMxCNT**: HRTIM_CNTxR register update enable timer x
Refer to TIMxCR description.

Bit 2 **TIMxDIER**: HRTIM_TIMxDIER register update enable timer x
Refer to TIMxCR description.

Bit 1 **TIMxICR**: HRTIM_TIMxICR register update enable timer x
Refer to TIMxCR description.

Bit 0 **TIMxCR**: HRTIM_TIMxCR register update enable timer x
This bit defines if the master timer MCR register is part of the list of registers to be updated by the burst DMA.
0: HRTIM_TIMxCR register is not updated by burst DMA accesses
1: HRTIM_TIMxCR register is updated by burst DMA accesses

33.5.83 HRTIM burst DMA data register (HRTIM_BDMADR)

Address offset: 0x770

Reset value: 0x0000 0000

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

Bits 31:0 **BDMADR[31:0]**: Burst DMA data register

Write accesses to this register triggers:

– the copy of the data value into the registers enabled in BDTxUPR and BDMUPR register bits

– the increment of the register pointer to the next location to be filled

33.5.84 HRTIM ADC extended trigger register (HRTIM_ADCER)

Address offset: 0x778

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ADC10TRG[4:0]					ADC9TRG[4:0]					ADC8TRG[4:0]				
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ADC7TRG[4:0]					ADC6TRG[4:0]					ADC5TRG[4:0]				
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **ADC10TRG[4:0]**: ADC trigger 10 selection
This bit selects the ADC trigger 10 source.
Refer to ADC6TRG[4:0] description.

Bits 25:21 **ADC9TRG[4:0]**: ADC trigger 9 selection
This bit selects the ADC trigger 9 source.
Refer to ADC5TRG[4:0] description.

Bits 20:16 **ADC8TRG[4:0]**: ADC trigger 8 selection
This bit selects the ADC trigger 8 source.
Refer to ADC6TRG[4:0] description.

Bit 15 Reserved, must be kept at reset value.

Bits 14:10 **ADC7TRG[4:0]**: ADC trigger 7 selection
This bit selects the ADC trigger 7 source.
Refer to ADC5TRG[4:0] description.

Bits 9:5 **ADC6TRG[4:0]**: ADC trigger 6 selection
This bit selects the ADC trigger 6 source.
0: Trigger on master compare 1
1: Trigger on master compare 2
2: Trigger on master compare 3
3: Trigger on master compare 4
4: Trigger on master period
5: Trigger on external event 6
6: Trigger on external event 7
7: Trigger on external event 8
8: Trigger on external event 9
9: Trigger on external event 10
10: Trigger on timer A compare 2
11: Trigger on timer A compare 4
12: Trigger on timer A period
13: Trigger on timer B compare 2
14: Trigger on timer B compare 4
15: Trigger on timer B period
16: Trigger on timer C compare 2
17: Trigger on timer C compare 4
18: Trigger on timer C period
19: Trigger on timer C reset and counter roll-over
20: Trigger on timer D compare 2
21: Trigger on timer D compare 4
22: Trigger on timer D period
23: Trigger on timer D reset and counter roll-over
24: Trigger on timer E compare 2
25: Trigger on timer E compare 3
26: Trigger on timer E compare 4
27: Trigger on timer E reset and counter roll-over
28: Trigger on timer F compare 2
29: Trigger on timer F compare 3
30: Trigger on timer F compare 4
31: Trigger on timer F period

Bits 4:0 **ADC5TRG[4:0]**: ADC trigger 5 selection

This bit selects the ADC trigger 5 source.

0: Trigger on master compare 1
1: Trigger on master compare 2
2: Trigger on master compare 3
3: Trigger on master compare 4
4: Trigger on master period
5: Trigger on external event 1
6: Trigger on external event 2
7: Trigger on external event 3
8: Trigger on external event 4
9: Trigger on external event 5
10: Trigger on timer A compare 3
11: Trigger on timer A compare 4
12: Trigger on timer A period
13: Trigger on timer A reset and counter roll-over
14: Trigger on timer B compare 3
15: Trigger on timer B compare 4
16: Trigger on timer B period
17: Trigger on timer B reset and counter roll-over
18: Trigger on timer C compare 3
19: Trigger on timer C compare 4
20: Trigger on timer C period
21: Trigger on timer D compare 3
22: Trigger on timer D compare 4
23: Trigger on timer D period
24: Trigger on timer E compare 3
25: Trigger on timer E compare 4
26: Trigger on timer E period
27: Trigger on timer F compare 2
28: Trigger on timer F compare 3
29: Trigger on timer F compare 4
30: Trigger on timer F period
31: Trigger on timer F reset and counter roll-over

33.5.85 HRTIM ADC trigger update register (HRTIM_ADCUR)

Address offset: 0x77C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AD10USRC[2:0]			Res.	AD9USRC[2:0]		
									rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AD8USRC[2:0]			Res.	AD7USRC[2:0]			Res.	AD6USRC[2:0]			Res.	AD5USRC[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

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

Bits 22:20 **AD10USRC[2:0]**: ADC trigger 10 update source
Refer to AD5USRC[2:0] description.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **AD9USRC[2:0]**: ADC trigger 9 update source
Refer to AD5USRC[2:0] description.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **AD8USRC[2:0]**: ADC trigger 8 update source
Refer to AD5USRC[2:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **AD7USRC[2:0]**: ADC trigger 7 update source
Refer to AD5USRC[2:0] description.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **AD6USRC[2:0]**: ADC trigger 6 update source
Refer to AD5USRC[2:0] description.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **AD5USRC[2:0]**: ADC trigger 5 update source
These bits define the source which triggers the update of the ADC5TRG[4:0] bitfield in the HRTIM_ADCER register (transfer from preload to active register). It only defines the source timer. The precise condition is defined within the timer itself, with the BRSTDMA[1:0] bitfield in HRTIM_MCR or the UPDGAT[3:0] bitfield in HRTIM_TIMxCR register.
000: Master timer
001: Timer A
010: Timer B
011: Timer C
100: Timer D
101: Timer E
110: Timer F
111: Reserved

33.5.86 HRTIM ADC post scaler register 1 (HRTIM_ADCPS1)

Address offset: 0x780

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ADC5PSC[4:0]					Res.	ADC4PSC[4:0]					Res.	ADC3PSC[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3PSC[3:0]				Res.	ADC2PSC[4:0]					Res.	ADC1PSC[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **ADC5PSC[4:0]**: ADC 5 post scaler
Refer to ADC1PSC[4:0] description.

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **ADC4PSC[4:0]**: ADC 4 post scaler
Refer to ADC1PSC[4:0] description.

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **ADC3PSC[4:0]**: ADC 3 post scaler
Refer to ADC1PSC[4:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **ADC2PSC[4:0]**: ADC 2 post scaler
Refer to ADC1PSC[4:0] description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **ADC1PSC[4:0]**: ADC 1 post scaler
This bit selects the ADC 1 Post scaler ratio.

33.5.87 HRTIM ADC post scaler register 2 (HRTIM_ADCPS2)

Address offset: 0x784

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ADC10PSC[4:0]					Res.	ADC9PSC[4:0]					Res.	ADC8PSC[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC8PSC[3:0]				Res.	ADC7PSC[4:0]					Res.	ADC6PSC[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **ADC10PSC[4:0]**: ADC 10 post scaler
Refer to ADC1PSC[4:0] description.

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **ADC9PSC[4:0]**: ADC 9 post scaler
Refer to ADC1PSC[4:0] description.

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **ADC8PSC[4:0]**: ADC 8 post scaler
Refer to ADC1PSC[4:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **ADC7PSC[4:0]**: ADC 7 post scaler
Refer to ADC1PSC[4:0] description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **ADC6PSC[4:0]**: ADC 6 post scaler
Refer to ADC1PSC[4:0] description.

33.5.88 HRTIM fault input register 3 (HRTIM_FLTINR3)

Address offset: 0x788

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLT4RSTM	FLT4CRES	FLT4CNT[3:0]				FLT4BLKS	FLT4BLKE	FLT3RSTM	FLT3CRES	FLT3CNT[3:0]				FLT3BLKS	FLT3BLKE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT2RSTM	FLT2CRES	FLT2CNT[3:0]				FLT2BLKS	FLT2BLKE	FLT1RSTM	FLT1CRES	FLT1CNT[3:0]				FLT1BLKS	FLT1BLKE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLT4RSTM**: Fault 4 reset mode
Refer to FLT5RSTM description.

Bit 30 **FLT4CRES**: Fault 4 counter reset
Refer to FLT5CRES description.

Bits 29:26 **FLT4CNT[3:0]**: Fault 4 counter
Refer to FLT5CNT description.

Bit 25 **FLT4BLKS**: Fault 4 blanking source
Refer to FLT5BLKS description.

Bit 24 **FLT4BLKE**: Fault 4 blanking enable
Refer to FLT5BLKE description.

Bit 23 **FLT3RSTM**: Fault 3 reset mode
Refer to FLT5RSTM description.

Bit 22 **FLT3CRES**: Fault 3 counter reset
Refer to FLT5CRES description.

Bits 21:18 **FLT3CNT[3:0]**: Fault 3 counter
Refer to FLT5CNT description.

Bit 17 **FLT3BLKS**: Fault 3 blanking source
Refer to FLT5BLKS description.

Bit 16 **FLT3BLKE**: Fault 3 blanking enable
Refer to FLT5BLKE description.

Bit 15 **FLT2RSTM**: Fault 2 reset mode
Refer to FLT5RSTM description.

Bit 14 **FLT2CRES**: Fault 2 counter reset
Refer to FLT5CRES description.

Bits 13:10 **FLT2CNT[3:0]**: Fault 2 counter
Refer to FLT5CNT description.

Bit 9 **FLT2BLKS**: Fault 2 blanking source
Refer to FLT5BLKS description.

Bit 8 **FLT2BLKE**: Fault 2 blanking enable
Refer to FLT5BLKE description.

Bit 7 **FLT1RSTM**: Fault 1 reset mode
Refer to FLT5RSTM description.

Bit 6 **FLT1CRES**: Fault 1 counter reset
Refer to FLT5CRES description.

Bits 5:2 **FLT1CNT[3:0]**: Fault 1 counter
Refer to FLT5CNT description.

Bit 1 **FLT1BLKS**: Fault 1 blanking source
Refer to FLT5BLKS description.

Bit 0 **FLT1BLKE**: Fault 1 blanking enable
Refer to FLT5BLKE description.

33.5.89 HRTIM fault input register 4 (HRTIM_FLTINR4)

Address offset: 0x78C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLT8RSTM	FLT8CRES	FLT8CNT[3:0]				FLT8BLKS	FLT8BLKE	FLT7RSTM	FLT7CRES	FLT7CNT[3:0]				FLT7BLKS	FLT7BLKE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT6RSTM	FLT6CRES	FLT6CNT[3:0]				FLT6BLKS	FLT6BLKE	FLT5RSTM	FLT5CRES	FLT5CNT[3:0]				FLT5BLKS	FLT5BLKE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLT8RSTM**: Fault 8 reset mode
Refer to FLT5RSTM description.

Bit 30 **FLT8CRES**: Fault 8 counter reset
Refer to FLT5CRES description.

Bits 29:26 **FLT8CNT[3:0]**: Fault 8 counter
Refer to FLT5CNT description.

Bit 25 **FLT8BLKS**: Fault 8 blanking source
Refer to FLT5BLKS description.

Bit 24 **FLT8BLKE**: Fault 8 blanking enable
Refer to FLT5BLKE description.

Bit 23 **FLT7RSTM**: Fault 7 reset mode
Refer to FLT5RSTM description.

Bit 22 **FLT7CRES**: Fault 7 counter reset
Refer to FLT5CRES description.

Bits 21:18 **FLT7CNT[3:0]**: Fault 7 counter
Refer to FLT5CNT description.

Bit 17 **FLT7BLKS**: Fault 7 blanking source
Refer to FLT5BLKS description.

Bit 16 **FLT7BLKE**: Fault 7 blanking enable
Refer to FLT5BLKE description.

Bit 15 **FLT6RSTM**: Fault 6 reset mode
Refer to FLT5RSTM description.

Bit 14 **FLT6CRES**: Fault 6 counter reset
Refer to FLT5CRES description.

Bits 13:10 **FLT6CNT[3:0]**: Fault 6 counter
Refer to FLT5CNT description.

Bit 9 **FLT6BLKS**: Fault 6 blanking source
Refer to FLT5BLKS description.

Bit 8 **FLT6BLKE**: Fault 6 blanking enable
Refer to FLT5BLKE description.

Bit 7 **FLT5RSTM**: Fault 5 reset mode

This bit selects the FAULT5 counter reset mode

0: Fault 5 counter is reset on each reset / roll-over event

1: Fault 5 counter is reset on each reset / roll-over event only if no fault occurred during last counting period.

Note: This bitfield can be written only when FLT5E enable bit is reset.

Bit 6 **FLT5CRES**: Fault 5 counter reset

This bit resets the FAULT5 counter. It is set by software and reset by hardware.

0: No action

1: Fault 5 counter is reset

Bits 5:2 **FLT5CNT[3:0]**: Fault 5 counter

This bitfield selects the FAULT5 counter threshold. A fault is considered valid when the number of events is equal to the (FLT5CNT[3:0]+1) value.

Bit 1 **FLT5BLKS**: Fault 5 blanking source

The FTL5BLKS bit selects the FAULT5 blanking source (refer to *Table 358* for details).

0: Fault 5 reset-aligned blanking window

1: Fault 5 Moving blanking window

Note: This bitfield can be written only when FLT5E enable bit is reset.

Bit 0 **FLT5BLKE**: Fault 5 blanking enable

The FTL5BLKE bit selects the FAULT5 blanking mode. The blanking source is defined by the FLT5BLKS bit.

0: No blanking on fault 5

1: Fault 5 blanking mode

Note: This bitfield can be written only when FLT5E enable bit is reset

33.5.90 HRTIM fault input register 5 (HRTIM_FLTINR5)

Address offset: 0x790

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT8LCK	FLT8F[3:0]				FLT8SRC[0]	FLT8P	FLT8E	FLT7LCK	FLT7F[3:0]				FLT7SRC[0]	FLT7P	FLT7E
rwo	rw	rw	rw	rw	rw	rw	rw	rwo	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FLT8LCK**: Fault 8 lock
Refer to FLT7LCK description.

Bits 14:11 **FLT8F[3:0]**: Fault 8 filter
Refer to FLT7F[3:0] description.

Bit 10 **FLT8SRC[0]**: Fault 8 source bit 0
Refer to FLT7SRC[0] description.

Bit 9 **FLT8P**: Fault 8 polarity
Refer to FLT7P description.

Bit 8 **FLT8E**: Fault 8 enable
Refer to FLT7E description.

Bit 7 **FLT7LCK**: Fault 7 lock
The FLT7LCK bit modifies the write attributes of the fault programming bit, so that they are protected against spurious write accesses.
This bit is write-once. Once it has been set, it cannot be modified till the next system reset.
0: FLT7E, FLT7P, FLT7SRC[1:0], FLT7F[3:0] and FLT7BLK bits are read/write.
1: FLT7E, FLT7P, FLT7SRC[1:0], FLT7F[3:0] and FLT7BLK bits cannot be written (read-only mode)

Bits 6:3 **FLT7F[3:0]**: Fault 7 filter

This bitfield defines the frequency used to sample FLT7 input and the length of the digital filter applied to FLT7. The digital filter is made of an event counter in which N events are needed to validate a transition on the output.

0000: No filter, FLT7 acts asynchronously
0001: $f_{SAMPLING} = f_{HRTIM}$, N = 2
0010: $f_{SAMPLING} = f_{HRTIM}$, N = 4
0011: $f_{SAMPLING} = f_{HRTIM}$, N = 8
0100: $f_{SAMPLING} = f_{FLTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{FLTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{FLTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{FLTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{FLTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{FLTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{FLTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{FLTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{FLTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{FLTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{FLTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{FLTS}/32$, N = 8

Note: This bitfield can be written only when FLT7E enable bit is reset.
This bitfield cannot be modified when FLT7LOCK has been programmed.

Bit 2 **FLT7SRC[0]**: Fault 7 source bit 0

The FTL7SRC[1:0] bitfield selects the FAULT7 input source (refer to *Table 325* for connection details).

00: Fault 7 input is HRTIM_FLT7 input pin
01: Fault 7 input is connected to a COMPx output
10: Fault 7 input is EEV7_muxout input pin
01: hrtim_ext_trg7

Note: This bitfield can be written only when FLT7E enable bit is reset.

Bit 1 **FLT7P**: Fault 7 polarity

This bit selects the FAULT7 input polarity.

0: Fault 7 input is active low
1: Fault 7 input is active high

Note: This bitfield can be written only when FLT7E enable bit is reset.

Bit 0 **FLT7E**: Fault 7 enable

This bit enables the global FAULT7 input circuitry.

0: Fault 7 input disabled
1: Fault 7 input enabled

33.5.91 HRTIM timer external event control register 4 (HRTIM_EECR4)

Address offset: 0x7A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EE13FAST	EE13SNS[1]
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE13SNS[0]	EE13POL	EE13SRC[1:0]		EE12FAST	EE12SNS[1:0]		EE12POL	EE12SRC[1:0]		EE11FAST	EE11SNS[1:0]		EE11POL	EE11SRC[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 17 **EE13FAST**: External event 3 fast mode
Refer to EE11FAST description.

Bits 16:15 **EE13SNS[1:0]**: External event 3 sensitivity
Refer to EE11SNS[1:0] description.

Bit 14 **EE13POL**: External event 3 polarity
Refer to EE11POL description.

Bits 13:12 **EE13SRC[1:0]**: External event 3 source
This bitfield is combined with EE13SRC[2] bit. Refer to EE11SRC[1:0] description.

Bit 11 **EE12FAST**: External event 2 fast mode
Refer to EE11FAST description.

Bits 10:9 **EE12SNS[1:0]**: External event 2 sensitivity
Refer to EE11SNS[1:0] description.

Bit 8 **EE12POL**: External event 2 polarity
Refer to EE11POL description.

Bits 7:6 **EE12SRC[1:0]**: External event 2 source
This bitfield is combined with EE12SRC[2] bit. Refer to EE11SRC[1:0] description.

Bit 5 **EE11FAST**: External event 11 fast mode
0: External event 11 is re-synchronized by the HRTIM logic before acting on outputs, which adds a f_{HRTIM} clock-related latency
1: External event 11 is acting asynchronously on outputs (low latency mode)
Note: This bit must not be modified once the counter in which the event is used is enabled (TxCEN bit set).

Bits 4:3 **EE11SNS[1:0]**: External event 11 sensitivity
00: On active level defined by EE11POL bit
01: Rising edge, whatever EE11POL bit value
10: Falling edge, whatever EE11POL bit value
11: Both edges, whatever EE11POL bit value

Bit 2 **EE11POL**: External event 11 polarity
This bit is only significant if EE11SNS[1:0] = 00.
0: External event is active high
1: External event is active low

Note: This parameter cannot be changed once the timer x is enabled. It must be configured prior to setting EE11FAST bit.

Bits 1:0 **EE11SRC[1:0]**: External event 11 source
This bitfield selects the External event 11 source. This bitfield is combined with EE11SRC[2] bit. See *Table 326* for details.
000:hrtim_eev11_1
001:hrtim_eev11_2
010:hrtim_eev11_3
011:hrtim_eev11_4
100:hrtim_eev11_5
101:hrtim_eev11_6
110:hrtim_eev11_7
111:hrtim_eev11_8

Note: This parameter cannot be changed once the timer x is enabled. It must be configured prior to setting EE11FAST bit.

33.5.92 HRTIM timer external event control register 5 (HRTIM_EECR5)

Address offset: 0x7A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EE16SNS[1]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE16SNS[0]	EE16POL	EE16SRC[1:0]		Res.	EE15SNS[1:0]		EE15POL	EE15SRC[1:0]		Res.	EE14SNS[1:0]		EE14POL	EE14SRC[1:0]	
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:15 **EE16SNS[1:0]**: External event 16 sensitivity
Refer to EE14SNS[1:0] description.

Bit 14 **EE16POL**: External event 16 polarity
Refer to EE14POL description.

Bits 13:12 **EE16SRC[1:0]**: External event 16 source
This bitfield is combined with EE16SRC[2] bit. Refer to EE14SRC[1:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:9 **EE15SNS[1:0]**: External event 15 sensitivity
Refer to EE14SNS[1:0] description.

Bit 8 **EE15POL**: External event 15 polarity
Refer to EE14POL description.

Bits 7:6 **EE15SRC[1:0]**: External event 15 source
This bitfield is combined with EE15SRC[2] bit. Refer to EE14SRC[1:0] description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **EE14SNS[1:0]**: External event 14 sensitivity
00: On active level defined by EE14POL bit
01: Rising edge, whatever EE14POL bit value
10: Falling edge, whatever EE14POL bit value
11: Both edges, whatever EE14POL bit value

Bit 2 **EE14POL**: External event 14 polarity
This bit is only significant if EE14SNS[1:0] = 00.
0: External event is active high
1: External event is active low

Bits 1:0 **EE14SRC[1:0]**: External event 14 source
This bitfield is combined with EE14SRC[2] bit. This bitfield selects the External event 14 source. See *Table 326* for details.
000:hrtim_eev14_1
001:hrtim_eev14_2
010:hrtim_eev14_3
011:hrtim_eev14_4
100:hrtim_eev14_5
101:hrtim_eev14_6
110:hrtim_eev14_7
111:hrtim_eev14_8

33.5.93 HRTIM timer external event control register 6 (HRTIM_EECR6)

Address offset: 0x7A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE16F[3:0]				Res.	Res.	EE15F[3:0]				Res.	Res.	EE14F[3:0]			
rw	rw	rw	rw			rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **EE16F[3:0]**: External event 16 filter
Refer to EE14F[3:0] description.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:6 **EE15F[3:0]**: External event 15 filter
Refer to EE14F[3:0] description.

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **EE14F[3:0]**: External event 14 filter
This bitfield defines the frequency used to sample external event 6 input and the length of the digital filter applied to EEV14. The digital filter is made of a counter in which N valid samples are needed to validate a transition on the output.
0000: Filter disabled
0001: $f_{SAMPLING} = f_{HRTIM}$, N=2
0010: $f_{SAMPLING} = f_{HRTIM}$, N=4
0011: $f_{SAMPLING} = f_{HRTIM}$, N=8
0100: $f_{SAMPLING} = f_{EEVS}/2$, N=6
0101: $f_{SAMPLING} = f_{EEVS}/2$, N=8
0110: $f_{SAMPLING} = f_{EEVS}/4$, N=6
0111: $f_{SAMPLING} = f_{EEVS}/4$, N=8
1000: $f_{SAMPLING} = f_{EEVS}/8$, N=6
1001: $f_{SAMPLING} = f_{EEVS}/8$, N=8
1010: $f_{SAMPLING} = f_{EEVS}/16$, N=5
1011: $f_{SAMPLING} = f_{EEVS}/16$, N=6
1100: $f_{SAMPLING} = f_{EEVS}/16$, N=8
1101: $f_{SAMPLING} = f_{EEVS}/32$, N=5
1110: $f_{SAMPLING} = f_{EEVS}/32$, N=6
1111: $f_{SAMPLING} = f_{EEVS}/32$, N=8

33.5.94 HRTIM timer external event control register 7 (HRTIM_EECR7)

Address offset: 0x7AC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	EE16SRC[2]	Res.	EE15SRC[2]	Res.	EE14SRC[2]	Res.	EE13SRC[2]	Res.	EE12SRC[2]	Res.	EE11SRC[2]	Res.	EE10SRC[2]	Res.	EE9SRC[2]
	rw		rw		rw		rw		rw		rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EE8SRC[2]	Res.	EE7SRC[2]	Res.	EE6SRC[2]	Res.	EE5SRC[2]	Res.	EE4SRC[2]	Res.	EE3SRC[2]	Res.	EE2SRC[2]	Res.	EE1SRC[2]
	rw		rw		rw		rw		rw		rw		rw		rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **EE16SRC[2]**: External event 16 source - bit 2
Refer to EE16SRC[1:0] bitfield description

Bit 29 Reserved, must be kept at reset value.

Bit 28 **EE15SRC[2]**: External event 15 source - bit 2
Refer to EE15SRC[1:0] bitfield description

Bit 27 Reserved, must be kept at reset value.

Bit 26 **EE14SRC[2]**: External event 14 source - bit 2
Refer to EE14SRC[1:0] bitfield description

Bit 25 Reserved, must be kept at reset value.

Bit 24 **EE13SRC[2]**: External event 13 source - bit 2
Refer to EE13SRC[1:0] bitfield description

Bit 23 Reserved, must be kept at reset value.

Bit 22 **EE12SRC[2]**: External event 12 source - bit 2
Refer to EE12SRC[1:0] bitfield description

Bit 21 Reserved, must be kept at reset value.

Bit 20 **EE11SRC[2]**: External event 11 source - bit 2
Refer to EE11SRC[1:0] bitfield description

Bit 19 Reserved, must be kept at reset value.

Bit 18 **EE10SRC[2]**: External event 10 source - bit 2
Refer to EE10SRC[1:0] bitfield description

Bit 17 Reserved, must be kept at reset value.

Bit 16 **EE9SRC[2]**: External event 9 source - bit 2
Refer to EE9SRC[1:0] bitfield description

Bit 15 Reserved, must be kept at reset value.

Bit 14 **EE8SRC[2]**: External event 8 source - bit 2
Refer to EE8SRC[1:0] bitfield description

Bit 13 Reserved, must be kept at reset value.

Bit 12 **EE7SRC[2]**: External event 7 source - bit 2
Refer to EE7SRC[1:0] bitfield description

Bit 11 Reserved, must be kept at reset value.

Bit 10 **EE6SRC[2]**: External event 6 source - bit 2
Refer to EE6SRC[1:0] bitfield description

Bit 9 Reserved, must be kept at reset value.

Bit 8 **EE5SRC[2]**: External event 5 source - bit 2
Refer to EE5SRC[1:0] bitfield description

Bit 7 Reserved, must be kept at reset value.

Bit 6 **EE4SRC[2]**: External event 4 source - bit 2
Refer to EE4SRC[1:0] bitfield description

Bit 5 Reserved, must be kept at reset value.

Bit 4 **EE3SRC[2]**: External event 3 source - bit 2
Refer to EE3SRC[1:0] bitfield description

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EE2SRC[2]**: External event 2 source - bit 2
Refer to EE2SRC[1:0] bitfield description

Bit 1 Reserved, must be kept at reset value.

Bit 0 **EE1SRC[2]**: External event 1 source - bit 2
Refer to EE1SRC[1:0] bitfield description

33.5.95 HRTIM control register 3 (HRTIM_CR3)

Address offset: 0x7C0

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTFUDISA	RTEUDISA	RTDUDISA	RTCUDISA	RTBUDISA	RTAUDISA	RMUDISA
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **RTFUDISA**: Remote timer F update disable authorization
Refer to RTAUDISA description.

Bit 5 **RTEUDISA**: Remote timer E update disable authorization
Refer to RTAUDISA description

Bit 4 **RTDUDISA**: Remote timer D update disable authorization
Refer to RTAUDISA description.

Bit 3 **RTCUDISA**: Remote timer C update disable authorization
Refer to RTAUDISA description.

Bit 2 **RTBUDISA**: Remote timer B update disable authorization
Refer to RTAUDISA description.

Bit 1 **RTAUDISA**: Remote timer A update disable authorization
This bit is set and cleared by software to authorize another HRTIMER to disable Timer A's update events.
0: Remote update disable requests are ignored.
1: Remote update disable requests authorized.

Bit 0 **RMUDISA**: Remote master update disable authorization
This bit is set and cleared by software to authorize another HRTIMER to disable master timer's update events.
0: Remote update disable requests are ignored.
1: Remote update disable requests authorized.

33.5.96 HRTIM option register (HRTIM_OR)

Address offset: 0x7FC

Reset value: 0x0000 0000

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

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

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

Bit 0 **HSPBKSEL**: HSP break selection

This bit enables the HSP break to be connected to the hrtim_sys_flt input and ORed with other system level fault (see *Table 332: HRTIM1/HRTIM2 system fault sources selection* for details).

Condition: HRTIM1

0: hsp1_break_out0 disabled

1: hsp1_break_out0 enabled and connected to hrtim1_sys_flt

Condition: HRTIM2

0: hsp1_break_out2 disabled

1: hsp1_break_out2 enabled and connected to hrtim2_sys_flt

33.5.97 HRTIM register map

The tables below summarize the HRTIM registers mapping.

Table 364. HRTIM Register map and reset values – master timer

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**HRTIM_MCR**	BRSTDMA[1:0]		MREPU	Res.	PREEN	DACSYNC[1:0]		Res.	Res.	TFCEN	TECEN	TDCEN	TCCEN	TBCEN	TACEN	MCEN	SYNCSRC[1:0]		SYNCOUT[1:0]		SYNCSTRTM	SYNCRSTM	SYNCIN[1:0]		INTLVD[1:0]		HALF	RETRIG	CONT	CKPSC[2:0]		
	Reset value	0	0	0		0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	**HRTIM_MISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPD	SYNC	MREP	MCMP4	MCMP3	MCMP2	MCMP1
	Reset value																										0	0	0	0	0	0	0
0x008	**HRTIM_MICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDC	SYNCC	MREPC	MCMP4C	MCMP3C	MCMP2C	MCMP1C
	Reset value																										0	0	0	0	0	0	0
0x00C	**HRTIM_MDIER**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDDE	SYNCDE	MREPDE	MCMP4DE	MCMP3DE	MCMP2DE	MCMP1DE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDIE	SYNCIE	MREPIE	MCMP4IE	MCMP3IE	MCMP2IE	MCMP1IE
	Reset value										0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x010	**HRTIM_MCNTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	**HRTIM_MPER**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPER[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0x018	**HRTIM_MREP**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MREP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	**HRTIM_MCMP1R**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	Reserved																																
0x024	**HRTIM_MCMP2R**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**HRTIM_MCMP3R**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	**HRTIM_MCMP4R**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This register can be preloaded (see *Table 350 on page 1599*).

Table 365. HRTIM register map and reset values – TIMx (x= A..F)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x100 (TIMA) 0x200 (TIMB) 0x300 (TIMC) 0x400 (TIMD) 0x500 (TIME) 0x600 (TIMF)	**HRTIM_ TIMxCR**	UPDGAT [3:0]				PREEN	DACSYNC[1:0]		MSTU	TEU	TDU	TCU	TBU	TAU	TxRSTU	TxREPU	TFU	DELCMP4[1:0]		DELCMP2[1:0]		SYNCSTRTx	SYNCRSTx	RSYNCU	INTLVD[1:0]		PSHPLL	HALF	RETRIG	CONT	CKPSCx[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104 (TIMA) 0x204 (TIMB) 0x304 (TIMC) 0x404 (TIMD) 0x504 (TIME) 0x604 (TIMF)	**HRTIM_ TIMxISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COTS	COS	O2CPY	O1CPY	O2STAT	O1STAT	IPPSTAT	CPPSTAT	CORDE	DLYPRT	RST	RSTx2	SETx2	RSTx1	SETx1	CPT2	CPT1	UPD	COT	REP	CMP4	CMP3	CMP2	CMP1
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108 (TIMA) 0x208 (TIMB) 0x308 (TIMC) 0x408 (TIMD) 0x508 (TIME) 0x608 (TIMF)	**HRTIM_ TIMxICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CORDEC	DLYPRTC	RSTC	RSTx2C	SET2xC	RST1xC	SET1xC	CPT2C	CPT1C	UPDC	COTC	REPC	CMP4C	CMP3C	CMP2C	CMP1C
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10C (TIMA) 0x20C (TIMB) 0x30C (TIMC) 0x40C (TIMD) 0x50C (TIME) 0x60C (TIMF)	**HRTIM_ TIMxDIER**(1)	CORDEDE	DLYPRTDE	RSTDE	RSTx2DE	SETx2DE	RSTx1DE	SET1xDE	CPT2DE	CPT1DE	UPDDE	COTDE	REPDE	CMP4DE	CMP3DE	CMP2DE	CMP1DE	CORDEIE	DLYPRTIE	RSTIE	RSTx2IE	SETx2IE	RSTx1IE	SET1xIE	CPT2IE	CPT1IE	UPDIE	COTIE	REPIE	CMP4IE	CMP3IE	CMP2IE	CMP1IE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x110 (TIMA) 0x210 (TIMB) 0x310 (TIMC) 0x410 (TIMD) 0x510 (TIME) 0x610 (TIMF)	**HRTIM_ CNTxR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNTx[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x114 (TIMA) 0x214 (TIMB) 0x314 (TIMC) 0x414 (TIMD) 0x514 (TIME) 0x614 (TIMF)	**HRTIM_ PERxR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERx[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0x118 (TIMA) 0x218 (TIMB) 0x318 (TIMC) 0x418 (TIMD) 0x518 (TIME) 0x618 (TIMF)	**HRTIM_ REPxR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x11C (TIMA) 0x21C (TIMB) 0x31C (TIMC) 0x41C (TIMD) 0x51C (TIME) 0x61C (TIMF)	**HRTIM_ CMP1xR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP1x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x120 (TIMA) 0x220 (TIMB) 0x320 (TIMC) 0x420 (TIMD) 0x520 (TIME) 0x620 (TIMF)	**HRTIM_ CMP1CxR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[7:0]								CMP1x[15:0]															
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x124 (TIMA) 0x224 (TIMB) 0x324 (TIMC) 0x424 (TIMD) 0x524 (TIME) 0x624 (TIMF)	**HRTIM_ CMP2xR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP2x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 365. HRTIM register map and reset values – TIMx (x= A..F) (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x128 (TIMA) 0x228 (TIMB) 0x328 (TIMC) 0x428 (TIMD) 0x528 (TIME) 0x628 (TIMF)	**HRTIM_CMP3xR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP3x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x12C (TIMA) 0x22C (TIMB) 0x32C (TIMC) 0x42C (TIMD) 0x52C (TIME) 0x62C (TIMF)	**HRTIM_CMP4xR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP4x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x130 (TIMA) 0x230 (TIMB) 0x330 (TIMC) 0x430 (TIMD) 0x530 (TIME) 0x630 (TIMF)	**HRTIM_CPT1xR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIR	CPT1x[15:0]															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x134 (TIMA) 0x234 (TIMB) 0x334 (TIMC) 0x434 (TIMD) 0x534 (TIME) 0x634 (TIMF)	**HRTIM_CPT2xR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIR	CPT2x[15:0]															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x138 (TIMA) 0x238 (TIMB) 0x338 (TIMC) 0x438 (TIMD) 0x538 (TIME) 0x638 (TIMF)	**HRTIM_DTxR**(1)	DTFLKx	DTFSLKx	Res.	Res.	DTM[1:0]		SDTFx	DTFx[8:0]									DTRLKx	DTRSLKx	Res.	DTPRSC[2:0]			SDTRx	DTRx[8:0]								
	Reset value	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x13C (TIMA) 0x23C (TIMB) 0x33C (TIMC) 0x43C (TIMD) 0x53C (TIME) 0x63C (TIMF)	**HRTIM_SETx1R**(1)	UPDATE	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x140 (TIMA) 0x240 (TIMB) 0x340 (TIMC) 0x440 (TIMD) 0x540 (TIME) 0x640 (TIMF)	**HRTIM_RSTx1R**(1)	UPDATE	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x144 (TIMA) 0x244 (TIMB) 0x344 (TIMC) 0x444 (TIMD) 0x544 (TIME) 0x644 (TIMF)	**HRTIM_SETx2R**(1)	UPDATE	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x148 (TIMA) 0x248 (TIMB) 0x348 (TIMC) 0x448 (TIMD) 0x548 (TIME) 0x648 (TIMF)	**HRTIM_RSTx2R**(1)	UPDATE	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14C (TIMA) 0x24C (TIMB) 0x34C (TIMC) 0x44C (TIMD) 0x54C (TIME) 0x64C (TIMF)	**HRTIM_EEFxR1**	Res.	Res.	EEEFLTR[4:0]					EEELTCH	EEDFLTR[4:0]					EEDLTCH	EECFLTR[4:0]					EECLTCH	EEBFLTR[4:0]					EEBLTCH	EEAFLTR[4:0]					EEALTCH
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 365. HRTIM register map and reset values – TIMx (x= A..F) (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x150 (TIMA) 0x250 (TIMB) 0x350 (TIMC) 0x450 (TIMD) 0x550 (TIME) 0x650 (TIMF)	**HRTIM_ EEFxR2**	Res.	Res.	EEJFLTR[3:0]					EEJLTCH	EEIFLTR[3:0]					EEILTCH	EEHFLTR[3:0]					EEHLTCH	EEGFLTR[3:0]					EEGLTCH	EEFFLTR[3:0]					EEFLTCH
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x154	**HRTIM_ RSTAR(1)**	TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x254	**HRTIM_ RSTBR(1)**	TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x354	**HRTIM_ RSTCR(1)**	TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x454	**HRTIM_ RSTDR(1)**	TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x554	**HRTIM_ RSTER(1)**	TIMFCMP2	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x654	**HRTIM_ RSTFR(1)**	TIMECMP2	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMECMP1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x158 (TIMA) 0x258 (TIMB) 0x358 (TIMC) 0x458 (TIMD) 0x558 (TIME) 0x658 (TIMF)	**HRTIM_ CHPxR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRTPW [3:0]				CARDT Y[2:0]			CARFRQ [3:0]			
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x15C	**HRTIM_ CPT1ACR**	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x25C	**HRTIM_ CPT1BCR**	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TFCMP2	TFCMP1	TF1RST	TA1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 365. HRTIM register map and reset values – TIMx (x= A..F) (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x35C	HRTIM_ CPT1CCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x45C	HRTIM_ CPT1DCR	TECMP2	TECMP1	TE1RST	TE1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x55C	HRTIM_ CPT1ECR	TFCMP2	TFCMP1	TF1RST	TF1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x65C	HRTIM_ CPT1FCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x160	HRTIM_ CPT2ACR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x260	HRTIM_ CPT2BCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x360	HRTIM_ CPT2CCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x460	HRTIM_ CPT2DCR	TECMP2	TECMP1	TE1RST	TE1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x560	HRTIM_ CPT2ECR	TFCMP2	TFCMP1	TF1RST	TF1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x660	HRTIM_ CPT2FCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 365. HRTIM register map and reset values – TIMx (x= A..F) (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x164 (TIMA) 0x264 (TIMB) 0x364 (TIMC) 0x464 (TIMD) 0x564 (TIME) 0x664 (TIMF)	**HRTIM_ OUTxR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIDL2	CHP2	FAULT2[1:0]		IDLES2	IDLEM2	POL2	Res.	Res.	Res.	Res.	DLYPRT[2:0]		DLYPRTEN	DTEN	DIDL1	CHP1	FAULT1[1:0]		IDLES1	IDLEM1	POL1	Res.
	Reset value									0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	
0x168 (TIMA) 0x268 (TIMB) 0x368 (TIMC) 0x468 (TIMD) 0x568 (TIME) 0x668 (TIMF)	**HRTIM_ FLTxR**	FLTLCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT7A8EN	FLT6A7EN	FLT5A6EN	FLT4A5EN	FLT3A4EN	FLT2A3EN	FLT1A2EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8EN	FLT7EN	FLT6EN	FLT5EN	FLT4EN	FLT3EN	FLT2EN	FLT1EN
	Reset value	0									0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x16C (TIMA) 0x26C(TIMB) 0x36C (TIMC) 0x46C (TIMD) 0x56C (TIME) 0x66C (TIMF)	**HRTIM_ TIMxCR2**	Res.	Res.	Res.	Res.	DCDCOE	COM	COR	COE	ADCTRGCH2	ADCTRGCH1	TRGHLFT	TRGHLF	Res.	Res.	GTCMP3	GTCMP1	FE ROM [1:0]		BM ROM [1:0]		AD ROM [1:0]		OUT ROM [1:0]		ROM [1:0]		Res.	UDM	Res.	DCDR	DCDS	DCDE
	Reset value					0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0
0x170 (TIMA) 0x270 (TIMB) 0x370 (TIMC) 0x470 (TIMD) 0x570 (TIME) 0x670 (TIMF)	**HRTIM_ TIMxEEFR3**	Res.	Res.	EEVZCNT[5:0]						EEVZ CNT[5:0]				Res.	EEVZRSTM	EEVZCRES	EEVZCE	Res.	Res.	EEVYCNT[5:0]					EEVYSEL [5:0]				Res.	EEVYRSTM	EEVYCRES	EEVYCE
	Reset value			0	0	0	0	0	0	0	0	0	0		0	0	0			0	0	0	0	0	0	0	0	0	0		0	0	0
0x174 (TIMA) 0x274 (TIMB) 0x374 (TIMC) 0x474 (TIMD) 0x574 (TIME) 0x674 (TIMF)	**HRTIM_ EESELxR**	Res.	Res.	EEVJSEL[1:0]		Res.	EEVISEL[1:0]		Res.	EEVHSEL[1:0]		Res.	EEVGSEL[1:0]	Res.	Res.	EEVFSEL[1:0]		Res.	Res.	EEVESEL[1:0]		Res.	EEVDSEL[1:0]		Res.	EEVCSEL[1:0]		Res.	EEVBSEL[1:0]		Res.	EEVASEL[1:0]	
	Reset value			0	0		0	0		0	0		0	0		0	0			0	0		0	0		0	0		0	0		0	0	0
0x178 (TIMA) 0x278 (TIMB) 0x378 (TIMC) 0x478 (TIMD) 0x578 (TIME) 0x678 (TIMF)	**HRTIM_ COTRGxR**	EXT_TRG5	EXT_TRG4	EXT_TRG3	EXT_TRG2	EXT_TRG1	EXT_TRG0	EEVJ	EEVI	EEVH	EEVG	EEVF	EEVE	EEVD	EEVC	EEVB	EEVA	FLT7A8	FLT6A7	FLT5A6	FLT4A5	FLT3A4	FLT2A3	FLT1A2	FLT8	FLT7	FLT6	FLT5	FLT4	FLT3	FLT2	FLT1	SWT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x17C (TIMA) 0x27C (TIMB) 0x37C (TIMC) 0x47C (TIMD) 0x57C (TIME) 0x67C (TIMF)	**HRTIM_ COFExR**	Res.	Res.	EEVJFRCB	EEVIFRCB	EEVHFRCB	EEVGFRCB	EEVFFRCB	EEVEFRCB	EEVDFRCB	EEVCFRCB	EEVBFRCB	EEVAFRCB	Res.	COFMB[2:0]			Res.	Res.	EEVJFRCA	EEVIFRCA	EEVHFRCA	EEVGFRCA	EEVFFRCA	EEVEFRCA	EEVDFRCA	EEVCFRCA	EEVBFRCA	EEVAFRCA	Res.	COFMA[2:0]		
	Reset value			0	0	0	0	0	0	0	0	0	0		0	0	0			0	0	0	0	0	0	0	0	0	0		0	0	0
0x180 (TIMA) 0x280 (TIMB) 0x380 (TIMC) 0x480 (TIMD) 0x580 (TIME) 0x680 (TIMF)	**HRTIM_ CODFxR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLY2FLTD[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLY1FLTD[8:0]								
	Reset value								0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0

1. This register can be preloaded (see *Table 350 on page 1599*).

Table 366. HRTIM register map and reset values – common functions

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x700	**HRTIM_CR1**	Res.	Res.	Res.	Res.	AD4USRC[2:0]			AD3USRC[2:0]			AD2USRC[2:0]			AD1USRC[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFUDIS	TEUDIS	TDUDIS	TCUDIS	TBUDIS	TAUDIS	MUDIS
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x704	**HRTIM_CR2**	Res.	Res.	Res.	Res.	Res.	DBSS [2:0]			Res.	Res.	SWPF	SWPE	SWPD	SWPC	SWPB	SWPA	Res.	TFRST	TERST	TDRST	TCRST	TBRST	TARST	MRST	Res.	TFSWU	TESWU	TDSWU	TCSWU	TBSWU	TASWU	MSWU
	Reset value						0	0	0			0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x708	**HRTIM_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8	FLT7	FLT6	SYSFLT	FLT5	FLT4	FLT3	FLT2	FLT1
	Reset value															0									0	0	0	0	0	0	0	0	0
0x70C	**HRTIM_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8C	FLT7C	FLT6C	SYSFLTC	FLT5C	FLT4C	FLT3C	FLT2C	FLT1C
	Reset value															0									0	0	0	0	0	0	0	0	0
0x710	**HRTIM_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8IE	FLT7IE	FLT6IE	SYSFLTIE	FLT5IE	FLT4IE	FLT3IE	FLT2IE	FLT1IE
	Reset value															0									0	0	0	0	0	0	0	0	0
0x714	**HRTIM_OENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TF2OEN	TF1OEN	TE2OEN	TE1OEN	TD2OEN	TD1OEN	TC2OEN	TC1OEN	TB2OEN	TB1OEN	TA2OEN	TA1OEN
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x718	**HRTIM_ODISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TF2ODIS	TF1ODIS	TE2ODIS	TE1ODIS	TD2ODIS	TD1ODIS	TC2ODIS	TC1ODIS	TB2ODIS	TB1ODIS	TA2ODIS	TA1ODIS
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x71C	**HRTIM_ODSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TF2ODS	TF1ODS	TE2ODS	TE1ODS	TD2ODS	TD1ODS	TC2ODS	TC1ODS	TB2ODS	TB1ODS	TA2ODS	TA1ODS
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x720	**HRTIM_BMCR**	BMSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFBM	TEBM	TDBM	TCBM	TBBM	TABM	MTBM	Res.	Res.	Res.	Res.	Res.	BMPREN	BMPRSC[3:0]				BMCLK[3:0]				BMOM	BME
	Reset value	0									0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x724	**HRTIM_ BMTRGR**	OCHPEV	EEV8	EEV7	TDEEV8	TAEEV7	TECMP2	TECMP1	TEREP	TFCMP1	TDCMP2	TFREP	TDREP	TDRST	TFRST	TCCMP1	TCREP	TCRST	TBCMP2	TBCMP1	TBREP	TBRST	TACMP2	TACMP1	TAREP	TARST	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTREP	MSTRST	SW
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x728	**HRTIM_ BMCMPR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMCMP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 366. HRTIM register map and reset values – common functions (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x72C	**HRTIM_ BMPER(1)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPER[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x730	**HRTIM_EECR1**	Res.	Res.	EE5FAST	EE5SNS[1:0]		EE5POL	EE5SRC[1:0]		EE4FAST	EE4SNS[1:0]		EE4POL	EE4SRC[1:0]		EE3FAST	EE3SNS[1:0]		EE3POL	EE3SRC[1:0]		EE2FAST	EE2SNS[1:0]		EE2POL	EE2SRC[1:0]		EE1FAST	EE1SNS[1:0]		EE1POL	EE1SRC[1:0]	
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x734	**HRTIM_EECR2**	Res.	Res.	Res.	EE10SNS[1:0]		EE10POL	EE10SRC[1:0]		Res.	EE9SNS[1:0]		EE9POL	EE9SRC[1:0]		Res.	EE8SNS[1:0]		EE8POL	EE8SRC[1:0]		Res.	EE7SNS[1:0]		EE7POL	EE7SRC[1:0]		Res.	EE6SNS[1:0]		EE6POL	EE6SRC[1:0]	
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0
0x738	**HRTIM_EECR3**	EEVSD[1:0]		Res.	Res.	EE10F[3:0]				Res.	Res.	EE9F[3:0]				Res.	Res.	EE8F[3:0]				Res.	Res.	EE7F[3:0]				Res.	Res.	EE6F[3:0]			
	Reset value	0	0			0	0	0	0			0	0	0	0			0	0	0	0			0	0	0	0			0	0	0	0
0x73C	**HRTIM_ ADC1R(1)**	ADC1TEPER	ADC1TEC4	ADC1TEC3	ADC1TFRST	ADC1TDPER	ADC1TDC4	ADC1TDC3	ADC1TFPER	ADC1TCPER	ADC1TCC4	ADC1TCC3	ADC1TFC4	ADC1TBRST	ADC1TBPER	ADC1TBC4	ADC1TBC3	ADC1TFC3	ADC1TARST	ADC1TAPER	ADC1TAC4	ADC1TAC3	ADC1TFC2	ADC1EEV5	ADC1EEV4	ADC1EEV3	ADC1EEV2	ADC1EEV1	ADC1MPER	ADC1MC4	ADC1MC3	ADC1MC2	ADC1MC1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x740	**HRTIM_ ADC2R(1)**	ADC2TERST	ADC2TEC4	ADC2TEC3	ADC2TEC2	ADC2TDRST	ADC2TDPER	ADC2TDC4	ADC2TFPER	ADC2TDC2	ADC2TCRST	ADC2TCPER	ADC2TCC4	ADC2TFC4	ADC2TCC2	ADC2TBPER	ADC2TBC4	ADC2TFC3	ADC2TBC2	ADC2TAPER	ADC2TAC4	ADC2TFC2	ADC2TAC2	ADC2EEV10	ADC2EEV9	ADC2EEV8	ADC2EEV7	ADC2EEV6	ADC2MPER	ADC2MC4	ADC2MC3	ADC2MC2	ADC2MC1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x744	**HRTIM_ ADC3R(1)**	ADC3TEPER	ADC3TEC4	ADC3TEC3	ADC3TFRST	ADC3TDPER	ADC3TDC4	ADC3TDC3	ADC3TFPER	ADC3TCPER	ADC3TCC4	ADC3TCC3	ADC3TFC4	ADC3TBRST	ADC3TBPER	ADC3TBC4	ADC3TBC3	ADC3TFC3	ADC3TARST	ADC3TAPER	ADC3TAC4	ADC3TAC3	ADC3TFC2	ADC3EEV5	ADC3EEV4	ADC3EEV3	ADC3EEV2	ADC3EEV1	ADC3MPER	ADC3MC4	ADC3MC3	ADC3MC2	ADC3MC1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x748	**HRTIM_ ADC4R(1)**	ADC4TERST	ADC4TEC4	ADC4TEC3	ADC4TEC2	ADC4TDRST	ADC4TDPER	ADC4TDC4	ADC4TFPER	ADC4TDC2	ADC4TCRST	ADC4TCPER	ADC4TCC4	ADC4TFC4	ADC4TCC2	ADC4TBPER	ADC4TBC4	ADC4TFC3	ADC4TBC2	ADC4TAPER	ADC4TAC4	ADC4TFC2	ADC4TAC2	ADC4EEV10	ADC4EEV9	ADC4EEV8	ADC4EEV7	ADC4EEV6	ADC4MPER	ADC4MC4	ADC4MC3	ADC4MC2	ADC4MC1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x750	**HRTIM_ FLTINR1**	FLT4LCK	FLT4F[3:0]				FLT4SRC[0]	FLT4P	FLT4E	FLT3LCK	FLT3F[3:0]				FLT3SRC[0]	FLT3P	FLT3E	FLT2LCK	FLT2F[3:0]				FLT2SRC[0]	FLT2P	FLT2E	FLT1LCK	FLT1F[3:0]				FLT1SRC[0]	FLT1P	FLT1E
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 366. HRTIM register map and reset values – common functions (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x754	**HRTIM_ FLTINR2**	Res.	Res.	Res.	Res.	Res.	Res.	FLTSD[1:0]		FLT8SRC[1]	FLT7SRC[1]	FLT6SRC[1]	FLT5SRC[1]	FLT4SRC[1]	FLT3SRC[1]	FLT2SRC[1]	FLT1SRC[1]	FLT6LCK	FLT6F[3:0]				FLT6SRC[0]	FLT6P	FLT6E	FLT5LCK	FLT5F[3:0]				FLT5SRC[0]	FLT5P	FLT5E
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x758	**HRTIM_ BDMUPDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP4	MCMP3	MCMP2	MCMP1	MREP	MPER	MCNT	MDIER	MICR	MCR
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x75C	**HRTIM_ BDTAUPR**	Res.	Res.	Res.	Res.	Res.	TIMACODER	TIMACOFER	TIMACOTRGR	TIMAEESELR	TIMAEEFR3	TIMACR2	TIMAFLTR	TIMAOUTR	TIMACHPR	TIMARSTR	TIMAEEFR2	TIMAEEFR1	TIMARST2R	TIMASET2R	TIMARST1R	TIMASET1R	TIMADTxR	TIMACMP4	TIMACMP3	TIMACMP2	TIMACMP1	TIMAREP	TIMAPER	TIMACNT	TIMADIER	TIMAICR	TIMACR
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x760	**HRTIM_ BDTBUPR**	Res.	Res.	Res.	Res.	Res.	TIMBCODER	TIMBCOFER	TIMBCOTRGR	TIMBEESELR	TIMBEEFR3	TIMBCR2	TIMBFLTR	TIMBOUTR	TIMBCHPR	TIMBRSTR	TIMBEEFR2	TIMBEEFR1	TIMBRST2R	TIMBSET2R	TIMBRST1R	TIMBSET1R	TIMBDTxR	TIMBCMP4	TIMBCMP3	TIMBCMP2	TIMBCMP1	TIMBREP	TIMBPER	TIMBCNT	TIMBDIER	TIMBICR	TIMBCR
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x764	**HRTIM_ BDTCUPR**	Res.	Res.	Res.	Res.	Res.	TIMCCODER	TIMCCOFER	TIMCCOTRGR	TIMCEESELR	TIMCEEFR3	TIMCCR2	TIMCFLTR	TIMCOUTR	TIMCCHPR	TIMCRSTR	TIMCEEFR2	TIMCEEFR1	TIMCRST2R	TIMCSET2R	TIMCRST1R	TIMCSET1R	TIMCDTxR	TIMCCMP4	TIMCCMP3	TIMCCMP2	TIMCCMP1	TIMCREP	TIMCPER	TIMCCNT	TIMCDIER	TIMCICR	TIMCCR
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x768	**HRTIM_ BDTDUPR**	Res.	Res.	Res.	Res.	Res.	TIMDCODER	TIMDCOFER	TIMDCOTRGR	TIMDEESELR	TIMDEEFR3	TIMDCR2	TIMDFLTR	TIMDOUTR	TIMDCHPR	TIMDRSTR	TIMDEEFR2	TIMDEEFR1	TIMDRST2R	TIMDSET2R	TIMDRST1R	TIMDSET1R	TIMDDTxR	TIMDCMP4	TIMDCMP3	TIMDCMP2	TIMDCMP1	TIMDREP	TIMDPER	TIMDCNT	TIMDDIER	TIMDICR	TIMDCR
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x76C	**HRTIM_ BDTEUPR**	Res.	Res.	Res.	Res.	Res.	TIMECODER	TIMECOFER	TIMECOTRGR	TIMEEESELR	TIMEEEFR3	TIMECR2	TIMEFLTR	TIMEOUTR	TIMECHPR	TIMERSTR	TIMEEEFR2	TIMEEEFR1	TIMERST2R	TIMESET2R	TIMERST1R	TIMESET1R	TIMEDTxR	TIMECMP4	TIMECMP3	TIMECMP2	TIMECMP1	TIMEREP	TIMEPER	TIMECNT	TIMEDIER	TIMEICR	TIMECR
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x770	**HRTIM_ BDMADR**	BDMADR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x774	**HRTIM_ BDTFUPR**	Res.	Res.	Res.	Res.	Res.	TIMFCODER	TIMFCOFER	TIMFCOTRGR	TIMFEESELR	TIMFEEFR3	TIMFCR2	TIMFFLTR	TIMFOUTR	TIMFCHPR	TIMFRSTR	TIMFEEFR2	TIMFEEFR1	TIMFRST2R	TIMFSET2R	TIMFRST1R	TIMFSET1R	TIMFDTxR	TIMFCMP4	TIMFCMP3	TIMFCMP2	TIMFCMP1	TIMFREP	TIMFPER	TIMFCNT	TIMFDIER	TIMFICR	TIMFCR
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 366. HRTIM register map and reset values – common functions (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x778	**HRTIM_ ADCER**(1)	Res.	ADC10TRG[4:0]					ADC9TRG[4:0]					ADC8TRG[4:0]					Res.	ADC7TRG[4:0]					ADC6TRG[4:0]					ADC5TRG[4:0]				
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x77C	**HRTIM_ ADCUR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AD10U SRC [2:0]			Res.	AD9U SRC [2:0]			Res.	AD8U SRC [2:0]			Res.	AD7U SRC [2:0]			Res.	AD6U SRC [2:0]			Res.	AD5U SRC [2:0]		
	Reset value										0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0
0x780	**HRTIM_ADCPS1**	Res.	Res.	Res.	ADC5PSC[4:0]					Res.	ADC4PSC[4:0]					Res.	ADC3PSC[4:0]					Res.	ADC2PSC[4:0]					Res.	ADC1PSC[4:0]				
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
0x784	**HRTIM_ADCPS2**	Res.	Res.	Res.	ADC10PSC[4:0]					Res.	ADC9PSC[4:0]					Res.	ADC8PSC[4:0]					Res.	ADC7PSC[4:0]					Res.	ADC6PSC[4:0]				
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0
0x788	**HRTIM_ FLTINR3**	FLT4RSTM	FLT4CRES	FLT4CNT[3:0]				FLT4BLKS	FLT4BLKE	FLT3RSTM	FLT3CRES	FLT3CNT[3:0]				FLT3BLKS	FLT3BLKE	FLT2RSTM	FLT2CRES	FLT2CNT[3:0]				FLT2BLKS	FLT2BLKE	FLT1RSTM	FLT1CRES	FLT1CNT[3:0]				FLT1BLKS	FLT1BLKE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x78C	**HRTIM_ FLTINR4**	FLT8RSTM	FLT8CRES	FLT8CNT[3:0]				FLT8BLKS	FLT8BLKE	FLT7RSTM	FLT7CRES	FLT7CNT[3:0]				FLT7BLKS	FLT7BLKE	FLT6RSTM	FLT6CRES	FLT6CNT[3:0]				FLT6BLKS	FLT6BLKE	FLT5RSTM	FLT5CRES	FLT5CNT[3:0]				FLT5BLKS	FLT5BLKE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x790	HRTIM_ FLTINR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8LCK	FLT8F[3:0]				FLT8SRC[0]	FLT8P	FLT8E	FLT7LCK	FLT7F[3:0]				FLT7SRC[0]	FLT7P	FLT7E
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x794-0x79C	Reserved	Res.																															
0x7A0	HRTIM_EECR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EE13FAST	EE13SNS[1:0]		EE13POL	EE13SRC[1:0]		EE12FAST	EE12SNS[1:0]		EE12POL	EE12SRC[1:0]		EE11FAST	EE11SNS[1:0]		EE11POL	EE11SRC[1:0]	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7A4	HRTIM_EECR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EE16SNS[1:0]		EE16POL	EE16SRC[1:0]		Res.	EE15SNS[1:0]		EE15POL	EE15SRC[1:0]		Res.	EE14SNS[1:0]		EE14POL	EE14SRC[1:0]	
	Reset value																0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0
0x7A8	HRTIM_EECR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EE16F[3:0]				Res.	Res.	EE15F[3:0]				Res.	Res.	EE14F[3:0]				
	Reset value																	0	0	0	0			0	0	0	0			0	0	0	0

Table 366. HRTIM register map and reset values – common functions (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x7AC	HRTIM_EECR7	Res.	EE16SRC[2]	Res.	EE15SRC[2]	Res.	EE14SRC[2]	Res.	EE13SRC[2]	Res.	EE12SRC[2]	Res.	EE11SRC[2]	Res.	EE10SRC[2]	Res.	EE9SRC[2]	Res.	EE8SRC[2]	Res.	EE7SRC[2]	Res.	EE6SRC[2]	Res.	EE5SRC[2]	Res.	EE4SRC[2]	Res.	EE3SRC[2]	Res.	EE2SRC[2]	Res.	EE1SRC[2]
	Reset value		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0
0x7B0-0x7BC	Reserved	Res.																															
0x7C0	HRTIM_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTFUDISA	RTEUDISA	RTDUDISA	RTCUDISA	RTBUDISA	RTAUDISA	RMUDISA
	Reset value																										0	0	0	0	0	0	0
0x7FC	HRTIM_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSPBKSEL
	Reset value																																0

1. This register can be preloaded (see *Table 350 on page 1599*).

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

34 Advanced-control timers (TIM1/TIM8/TIM20)

34.1 TIM1/TIM8/TIM20 introduction

The advanced-control timers (TIM1/TIM8/TIM20) consist of a 16-bit autoreload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1/TIM8/TIM20) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 34.3.35: Timer synchronization*.

34.2 TIM1/TIM8/TIM20 main features

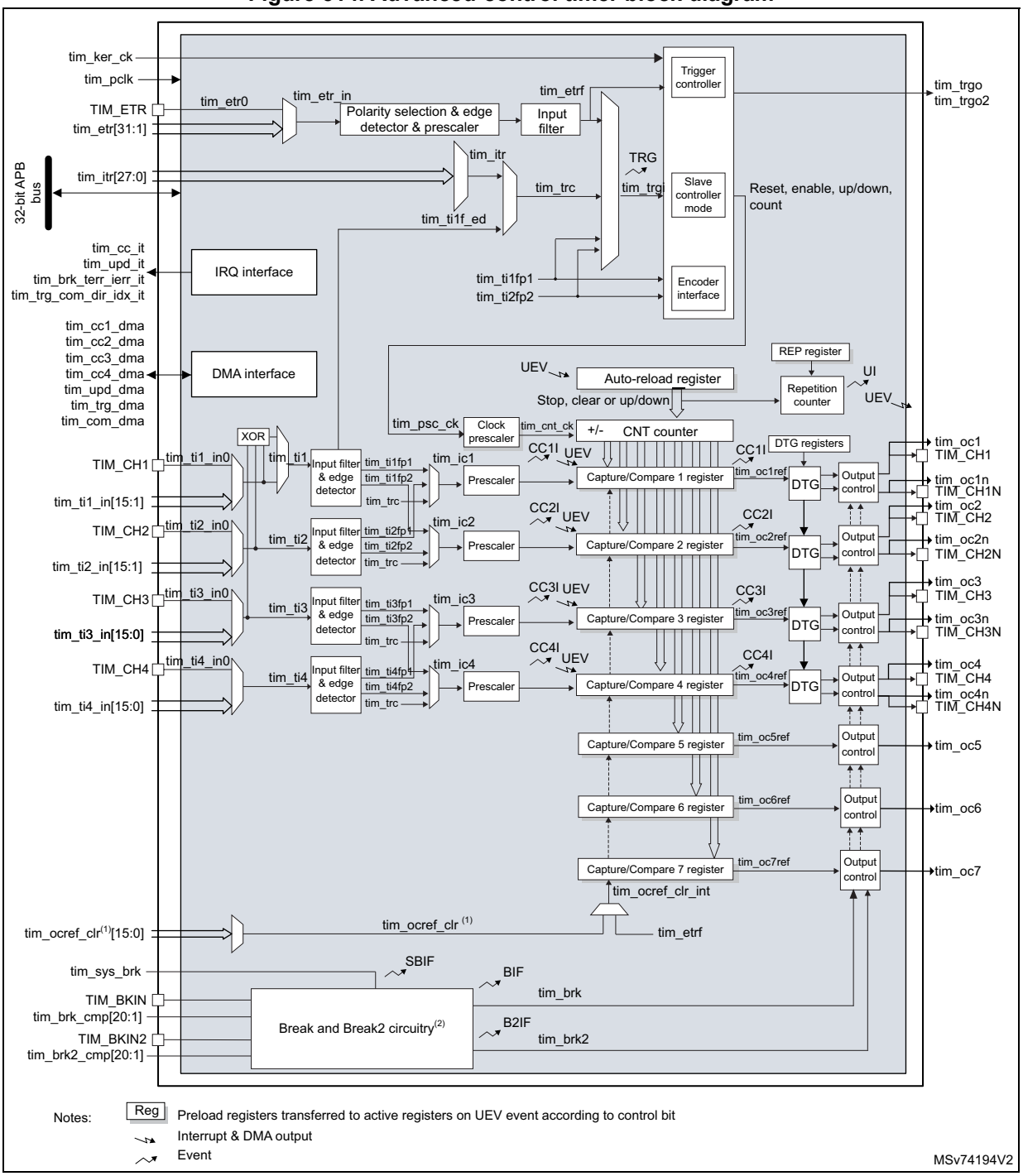
TIM1/TIM8/TIM20 timer features include:

- 16-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency by any factor from 1 to 65536.
- Seven independent channels for:
  - Input capture (but channels 5, 6 and 7)
  - Output compare
  - PWM generation (edge and center-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger)
  - Input capture
  - Output compare
- Incremental encoders, quadrature encoders and hall-sensors support
- Trigger input for external clock or cycle-by-cycle current management
- ADC synchronization for jitter-free sampling points

34.3 TIM1/TIM8/TIM20 functional description

34.3.1 Block diagram

Figure 514. Advanced-control timer block diagram



1. This feature is not available on all timers, refer to *Section 34.3.2: TIM1/TIM8/TIM20 pins and internal signals*.
2. See *Figure 565: Break and Break2 circuitry overview* for details.

34.3.2 TIM1/TIM8/TIM20 pins and internal signals

The tables in this section summarize the TIM inputs and outputs

Table 367. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/Output	Timer multi-purpose channels. Each channel can be used for capture, compare or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_CH1N TIM_CH2N TIM_CH3N TIM_CH4N	Output	Timer complementary outputs, derived from TIM_CHx outputs with the possibility to have deadtime insertion.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
TIM_BKIN TIM_BKIN2	Input / Output	Break and Break2 inputs. These inputs can also be configured in bidirectional mode.

Table 368. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[31:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulsewidth control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[27:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo/tim_trgo2	Output	Internal trigger outputs. These triggers are used by other timers and /or other peripherals.

Table 368. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_ocx (x = 1 to 4)	Output	Internal output compare signals of channels 1 to 4. These outputs can be used for triggering/synchronizing other internal peripherals.
tim_ocxn (x = 1 to 4)	Output	Internal timer complementary outputs, derived from tim_ocx internal outputs with the possibility to have deadtime insertion.
tim_ocx (x = 5 to 7)	Output	Internal output compare signals of channels 5 to 7. These outputs can be used for triggering/synchronizing other internal peripherals.
tim_ocref_clr[15:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[20:1]	Input	Break input for internal signals
tim_brk2_cmp[20:1]	Input	Break2 input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock
tim_cc_it	Output	Timer capture/compare interrupt
tim_upd_it	Output	Timer update event interrupt
tim_brk_terr_ierr_it	Output	Timer break, break2, transition error and index error interrupt
tim_trgi_com_dir_idx_it	Output	Timer trigger, commutation, direction and index interrupt
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture / compare 1..4 dma requests
tim_upd_dma	Output	Timer update dma request
tim_trgi_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

Table 369, *Table 370*, *Table 371*, and *Table 372* list the sources connected to the tim_ti[4:1] input multiplexers.

Table 369. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources		
	TIM1	TIM8	TIM20
tim_ti1_in0	TIM1_CH1	TIM8_CH1	TIM20_CH1
tim_ti1_in1	comp1_out		
tim_ti1_in2	comp2_out		
tim_ti1_in3	comp3_out		
tim_ti1_in4	comp4_out		
tim_ti1_in[15:8]	Reserved		

Table 370. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources		
	TIM1	TIM8	TIM20
tim_ti2_in0	TIM1_CH2	TIM8_CH2	TIM20_CH2
tim_ti2_in[15:1]	Reserved		

Table 371. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources		
	TIM1	TIM8	TIM20
tim_ti3_in0	TIM1_CH3	TIM8_CH3	TIM20_CH3
tim_ti3_in[15:1]	Reserved		

Table 372. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources		
	TIM1	TIM8	TIM20
tim_ti4_in0	TIM1_CH4	TIM8_CH4	TIM20_CH4
tim_ti4_in[15:1]	Reserved		

Table 373 lists the internal sources connected to the tim_itr input multiplexer.

Table 373. Internal trigger connection

Timer internal trigger input signal	TIM1	TIM8	TIM20
tim_itr0	Reserved	tim1_trgo	
tim_itr1		tim1_trgo2	
tim_itr2	tim2_trgo		

Table 373. Internal trigger connection (continued)

Timer internal trigger input signal	TIM1	TIM8	TIM20
tim_itr3	tim3_trgo		
tim_itr4	tim4_trgo		
tim_itr5	tim5_trgo		
tim_itr6	tim8_trgo	Reserved	tim8_trgo
tim_itr7	tim8_trgo2		tim8_trgo2
tim_itr8	tim9_trgo		
tim_itr9	tim10_oc1		
tim_itr10	tim11_oc1		
tim_itr11	tim12_trgo		
tim_itr12	tim13_oc1		
tim_itr13	tim14_oc1		
tim_itr14	tim15_trgo		
tim_itr15	tim16_oc1		
tim_itr16	tim17_oc1		
tim_itr17	tim20_trgo		Reserved
tim_itr18	tim20_trgo2		
tim_itr[20:19]	Reserved		
tim_itr21	hrtim1_out_sync2		
tim_itr22	hrtim2_out_sync2		
tim_itr[27:23]	Reserved		

Table 374 lists the internal sources connected to the tim_etr input multiplexer.

Table 374. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signals assignment		
	TIM1	TIM8	TIM20
tim_etr0	TIM1_ETR	TIM8_ETR	TIM20_ETR
tim_etr1	comp1_out		
tim_etr2	comp2_out		
tim_etr3	comp3_out		
tim_etr4	comp4_out		
tim_etr5	comp5_out		
tim_etr6	comp6_out		
tim_etr7	comp7_out		

Table 374. Interconnect to the tim_etr input multiplexer (continued)

Timer external trigger input signal	Timer external trigger signals assignment		
	TIM1	TIM8	TIM20
tim_etr8	comp8_out		
tim_etr[16:9]	Reserved		
tim_etr17	adc1_awd1	adc2_awd1	adc3_awd1
tim_etr18	adc1_awd2	adc2_awd2	adc3_awd2
tim_etr19	adc1_awd3	adc2_awd3	adc3_awd3
tim_etr20	adc2_awd1	adc3_awd1	adc4_awd1
tim_etr21	adc2_awd2	adc3_awd2	adc4_awd2
tim_etr22	adc2_awd3	adc3_awd3	adc4_awd3
tim_etr[31:23]	Reserved		

Table 375, *Table 376*, and *Table 378* list the sources connected to the tim_brk and tim_brk2 inputs.

Table 375. Interconnect to the tim_brk_cmp multiplexer

tim_brk inputs	TIM1	TIM8	TIM20
TIM_BKIN	TIM1_BKIN pin	TIM8_BKIN pin	TIM20_BKIN pin
tim_brk_cmp1	comp1_out		
tim_brk_cmp2	comp2_out		
tim_brk_cmp3	comp3_out		
tim_brk_cmp4	comp4_out		
tim_brk_cmp5	comp5_out		
tim_brk_cmp6	comp6_out		
tim_brk_cmp7	comp7_out		
tim_brk_cmp8	comp8_out		
tim_brk_cmp9	HRTIM1_FLT1		
tim_brk_cmp10	HRTIM1_FLT2		
tim_brk_cmp11	HRTIM2_FLT1		
tim_brk_cmp12	HRTIM2_FLT2		
tim_brk_cmp13	hsp1_brk_out0		
tim_brk_cmp14	hsp1_brk_out1		
tim_brk_cmp15	Reserved	TIM1_BKIN	
tim_brk_cmp16	TIM8_BKIN	Reserved	TIM8_BKIN
tim_brk_cmp17	TIM20_BKIN		Reserved

Table 375. Interconnect to the tim_brk_cmp multiplexer (continued)

tim_brk inputs	TIM1	TIM8	TIM20
tim_brk_cmp18	TIM15_BKIN		
tim_brk_cmp19	TIM16_BKIN		
tim_brk_cmp20	TIM17_BKIN		

Table 376. Timer break2 interconnect

tim_brk2 inputs	TIM1	TIM8	TIM20
TIM_BKIN2	TIM1_BKIN2 pin	TIM8_BKIN2 pin	TIM20_BKIN2 pin
tim_brk2_cmp1	comp1_out		
tim_brk2_cmp2	comp2_out		
tim_brk2_cmp3	comp3_out		
tim_brk2_cmp4	comp4_out		
tim_brk2_cmp5	comp5_out		
tim_brk2_cmp6	comp6_out		
tim_brk2_cmp7	comp7_out		
tim_brk2_cmp8	comp8_out		
tim_brk2_cmp[12:9]	Reserved		
tim_brk2_cmp13	hsp1_brk_out1		
tim_brk2_cmp14	hsp1_brk_out2		
tim_brk2_cmp15	Reserved	TIM1_BKIN	
tim_brk2_cmp16	TIM8_BKIN	Reserved	TIM8_BKIN
tim_brk2_cmp17	TIM20_BKIN		Reserved
tim_brk2_cmp[20:18]	Reserved		

Table 377. Interconnect to tim_ocref_clr multiplexer

tim_brk2 inputs	TIM1	TIM8	TIM20
tim_ocref_clr0	comp1_out		
tim_ocref_clr1	comp2_out		
tim_ocref_clr2	comp3_out		
tim_ocref_clr3	comp4_out		
tim_ocref_clr4	comp5_out		
tim_ocref_clr5	comp6_out		
tim_ocref_clr6	comp7_out		
tim_ocref_clr7	comp8_out		
tim_ocref_clr[15:8]	Reserved		

Table 378. System break interconnect

tim_sys_brk inputs	TIM1 / TIM8 / TIM20	Enable bit in SBS_CFGR2 register
tim_sys_brk0	Cortex®-M7 LOCKUP	CLL
tim_sys_brk1	SRAM double ECC error	SEL
tim_sys_brk2	Programmable Voltage Detector (PVD)	PVDL
tim_sys_brk3	Non-Volatile memory double ECC error	ECCL
tim_sys_brk4	Cortex®-M7 Data Cache ECC error	DCECCL
tim_sys_brk5	Clock Security System (CSS)	None (always enabled)

34.3.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software, even when the counter is running.

The time-base unit includes:

- Counter register (TIM_CNT)
- Prescaler register (TIM_PSC)
- Autoreload register (TIM_ARR)
- Repetition counter register (TIM_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIM_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIM_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

Note: An update overrun status bit (UIOVRF in TIM_SR register) indicates that an update request occurred while an update interrupt is pending (update interrupt flag set). This bit is useful when debugging closed loop applications, to indicate that:

- the update interrupt service routine is too long with respect to the interval between consecutive update events

- the update interrupt service routine was served too late with respect to the subsequent update event

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIM_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note: The counter starts counting 1 clock cycle after setting the CEN bit in the TIM_CR1 register.

Prescaler description

The prescaler divides the counter clock frequency by any factor from 1 to 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIM_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 515 and *Figure 516* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 515. Counter timing diagram with prescaler division change from 1 to 2

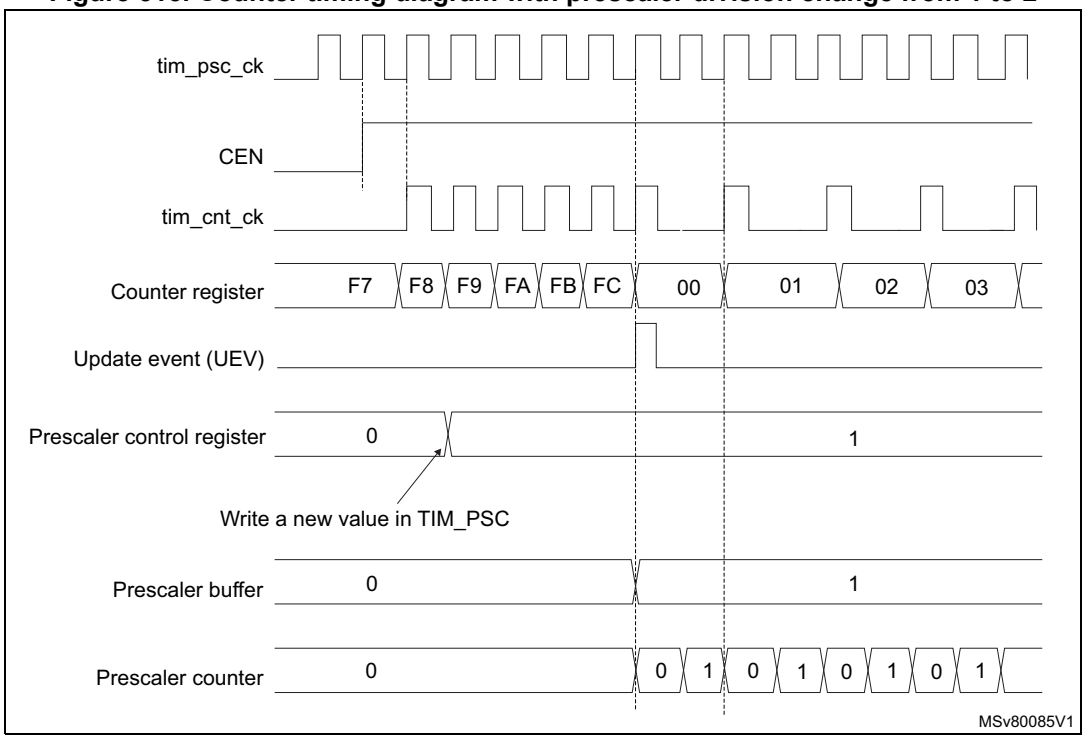
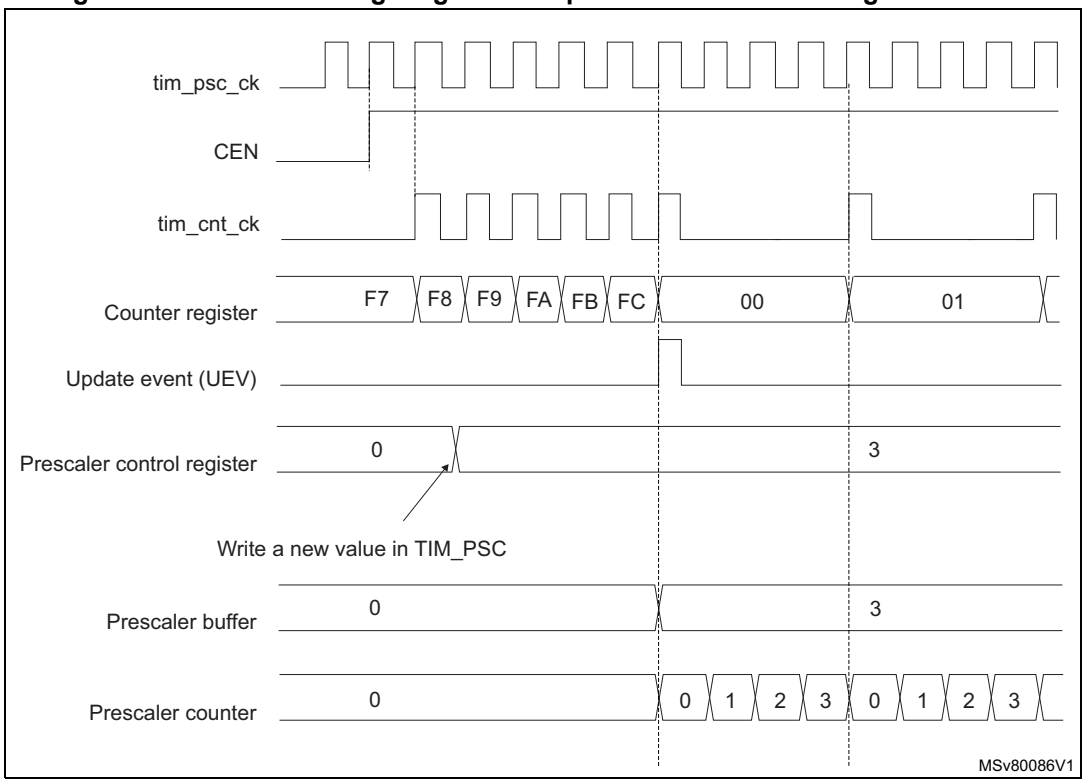


Figure 516. Counter timing diagram with prescaler division change from 1 to 4



34.3.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIM_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register,
- The autoreload shadow register is updated with the preload value (TIM_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 517. Counter timing diagram, internal clock divided by 1

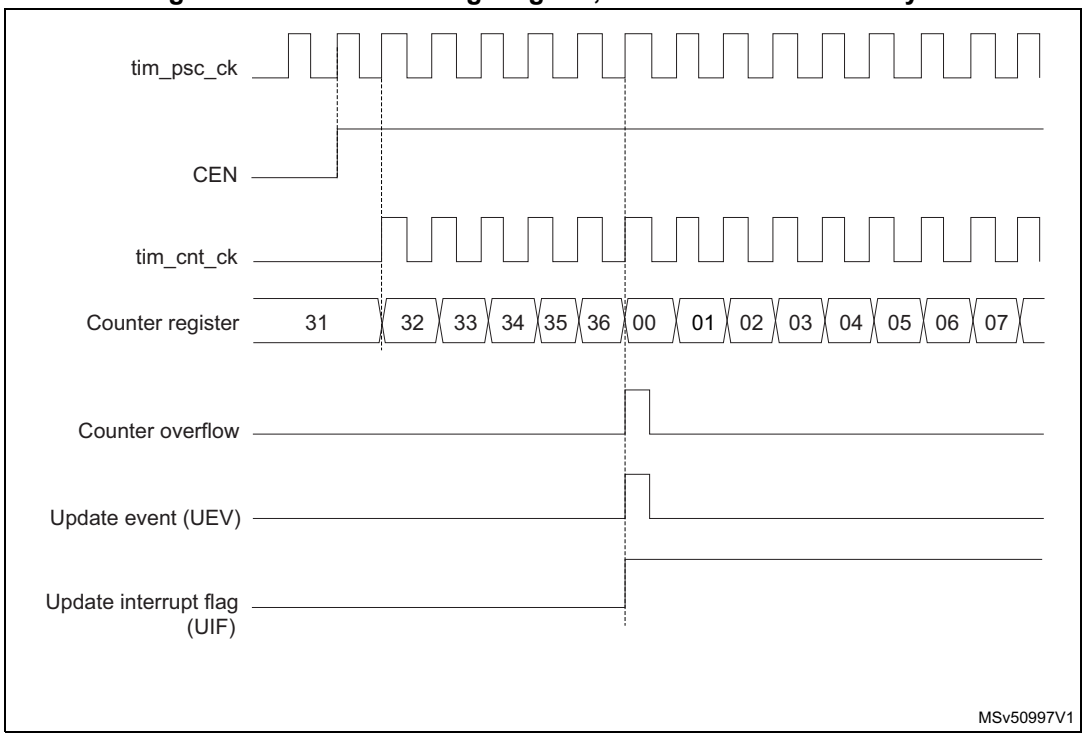


Figure 518. Counter timing diagram, internal clock divided by 2

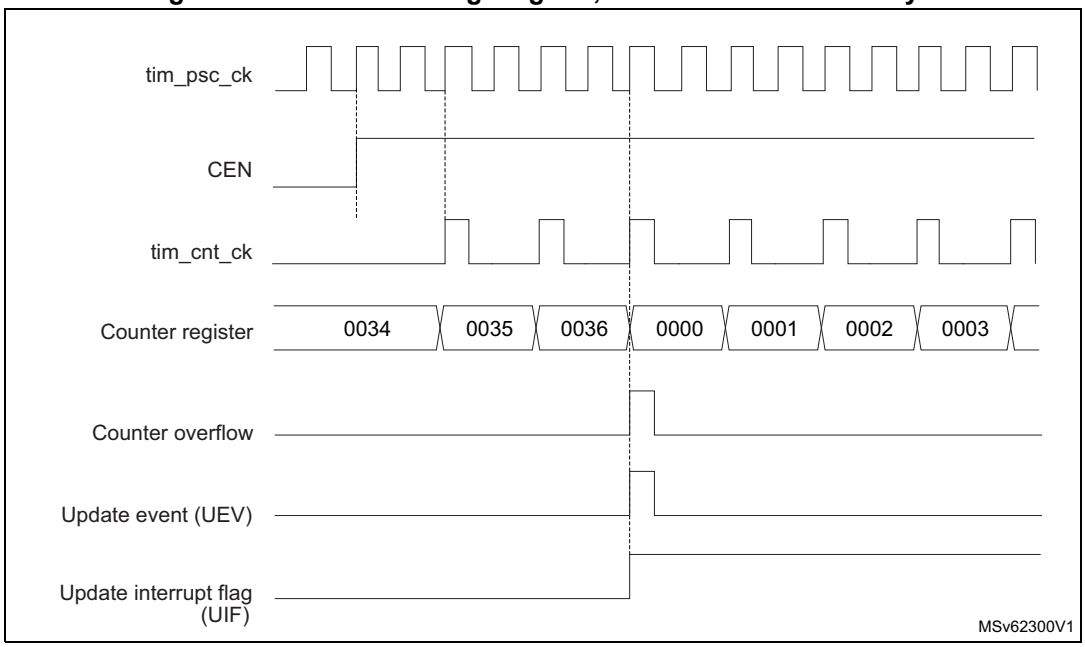


Figure 519. Counter timing diagram, internal clock divided by 4

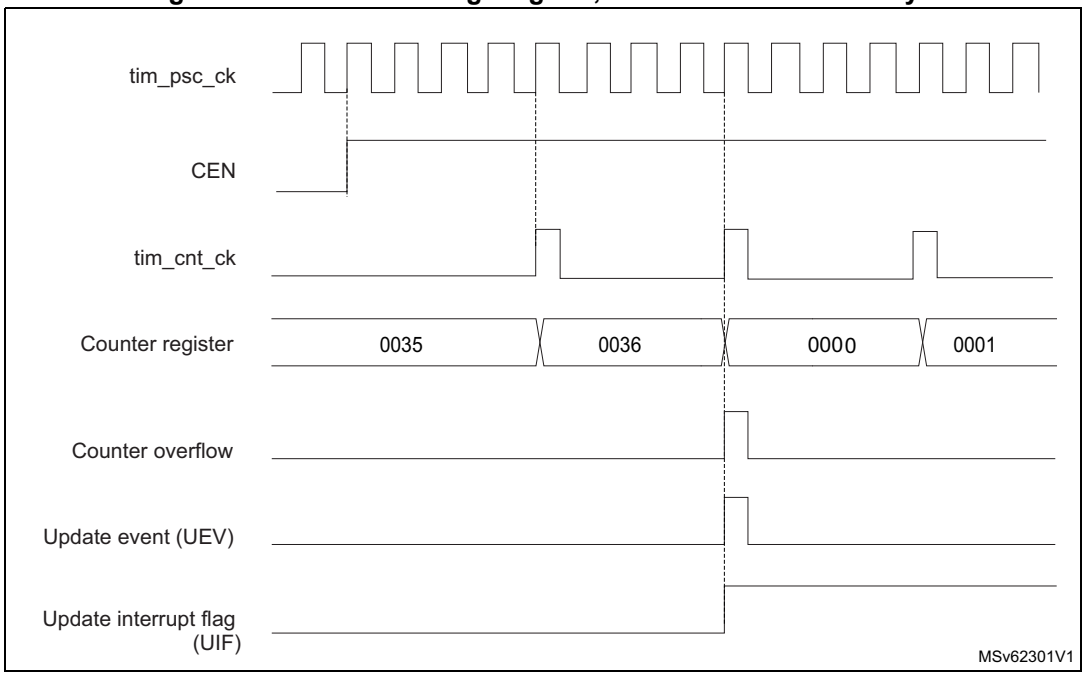


Figure 520. Counter timing diagram, internal clock divided by N

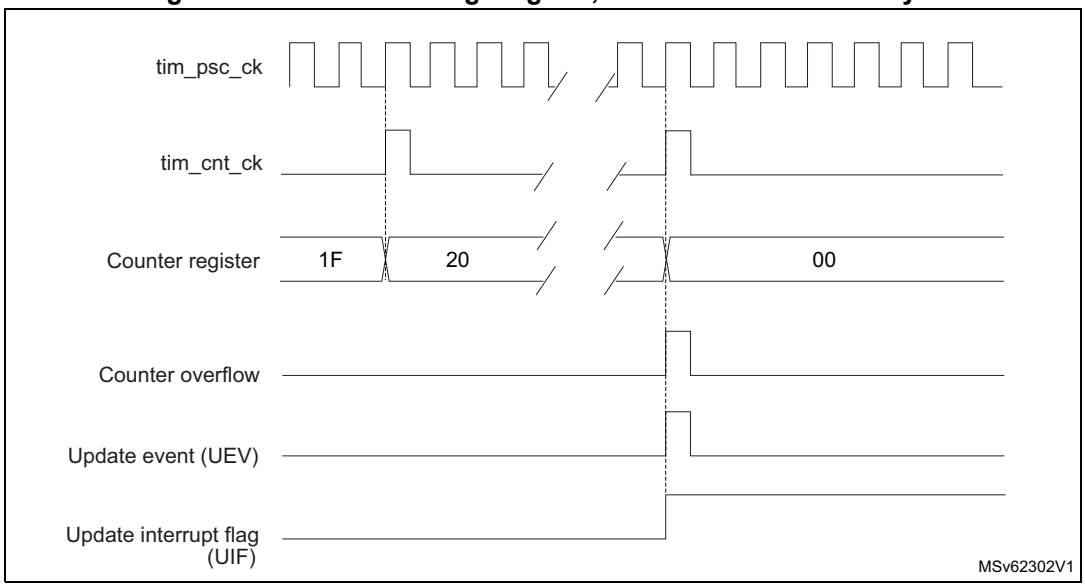


Figure 521. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded)

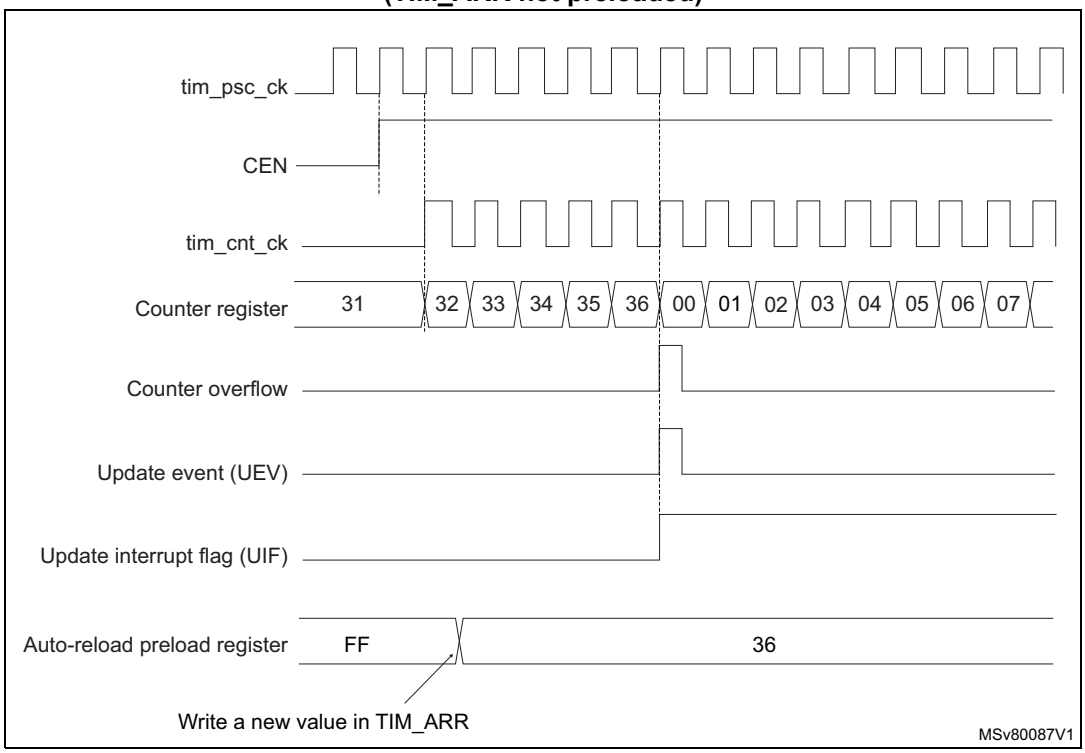
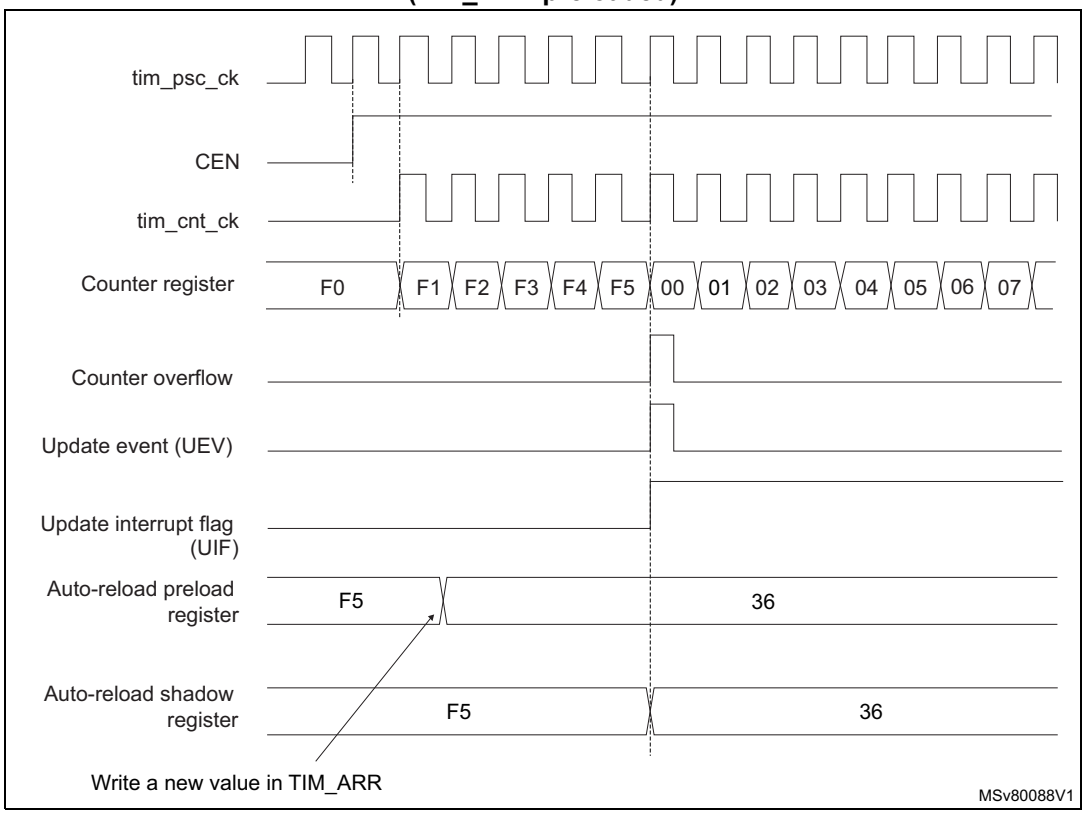


Figure 522. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the autoreload value (content of the TIM_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIM_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 523. Counter timing diagram, internal clock divided by 1

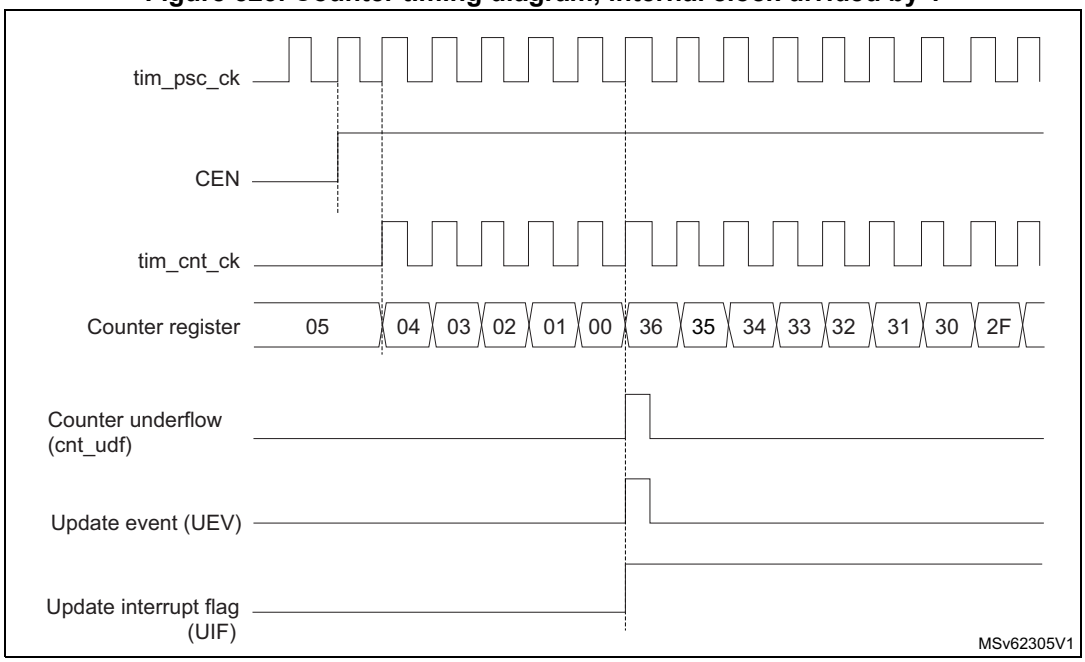


Figure 524. Counter timing diagram, internal clock divided by 2

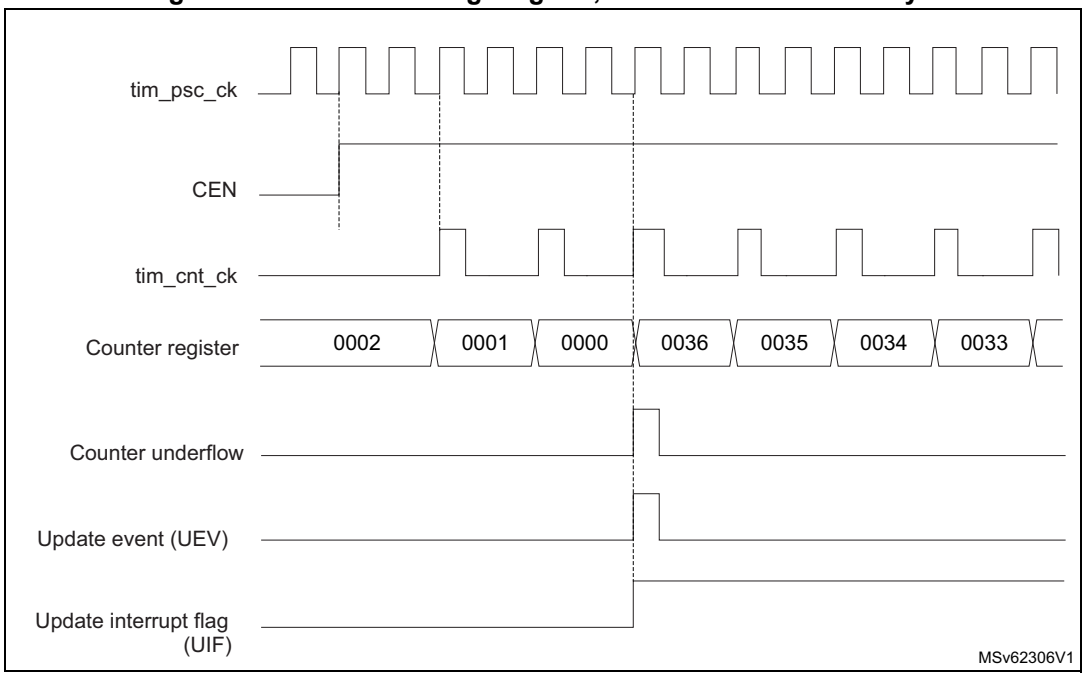


Figure 525. Counter timing diagram, internal clock divided by 4

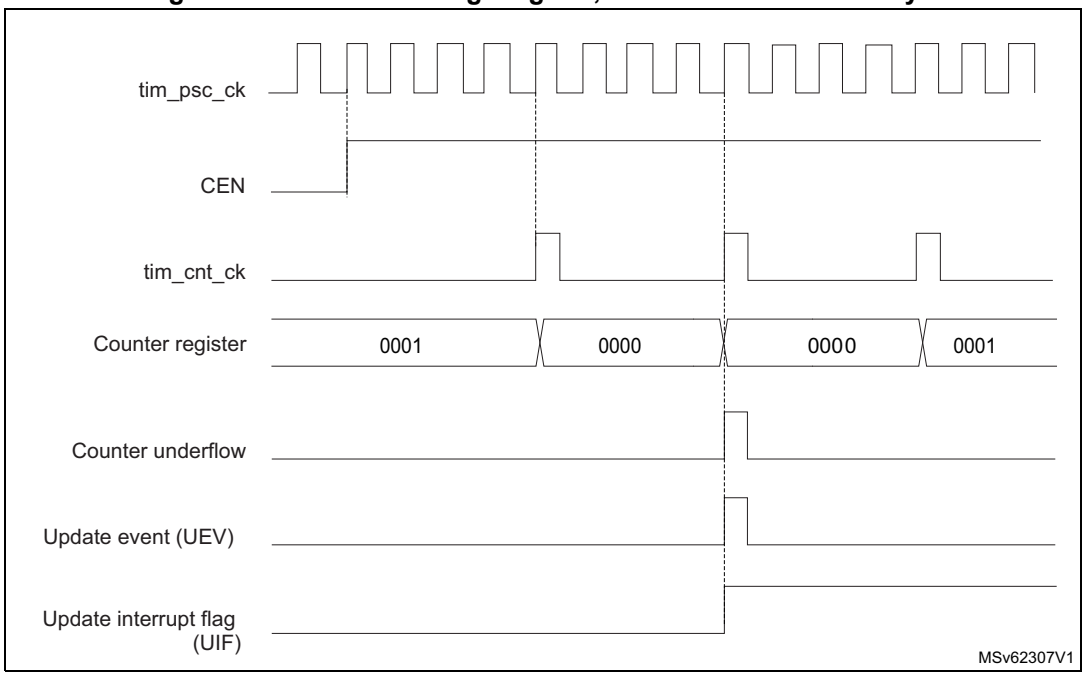


Figure 526. Counter timing diagram, internal clock divided by N

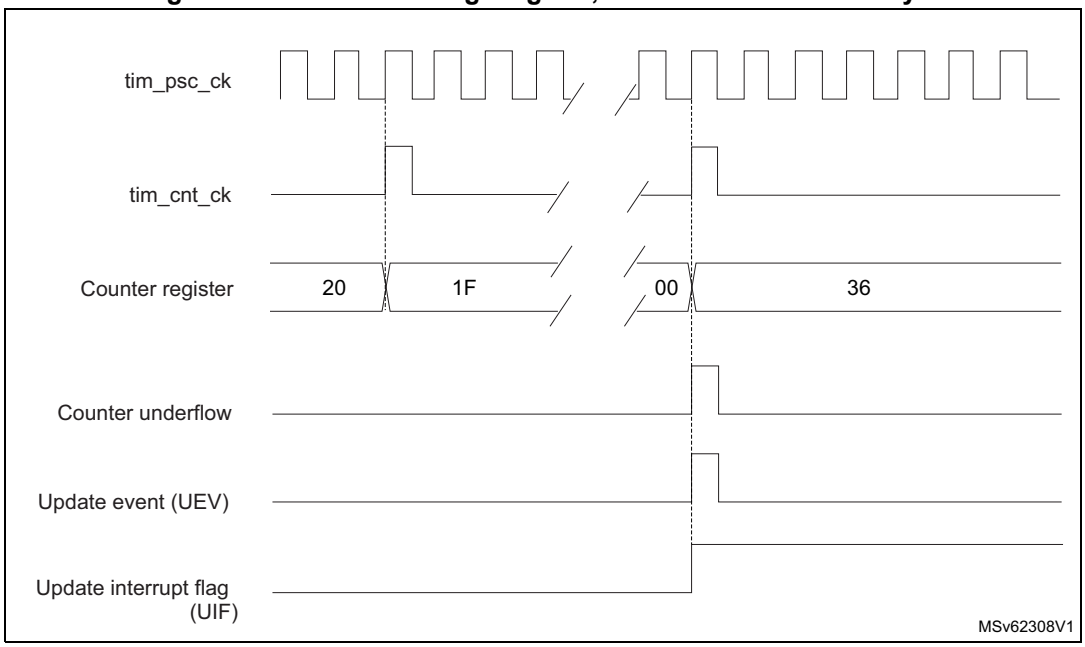
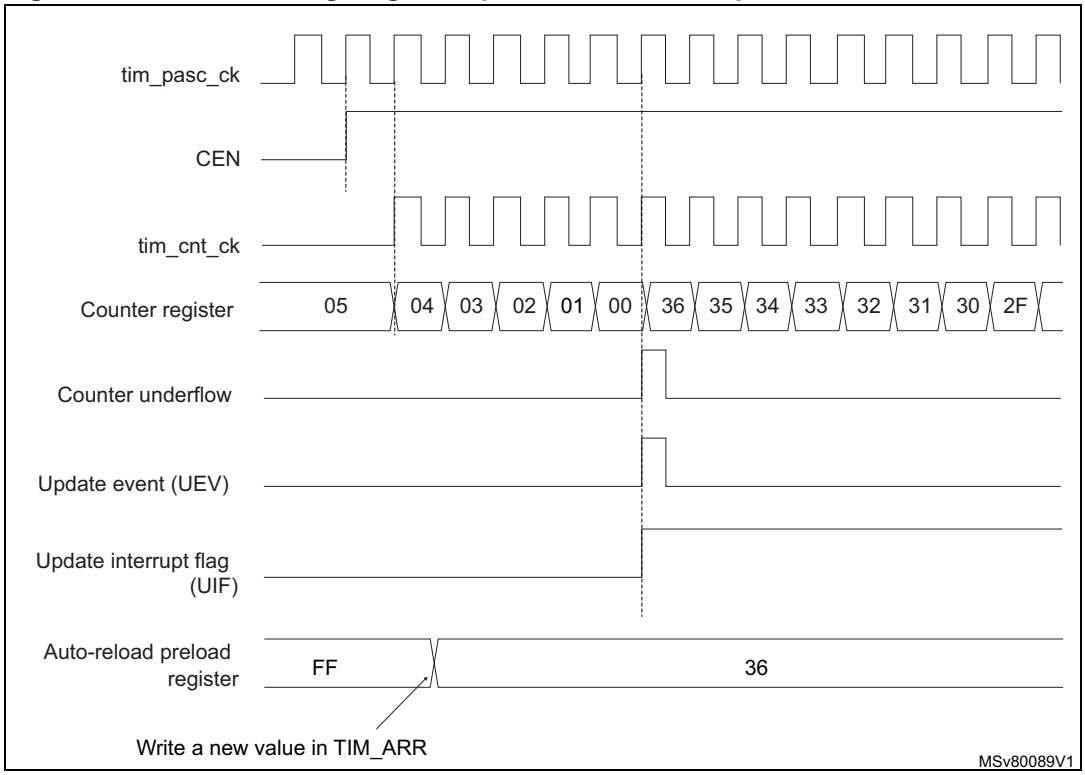


Figure 527. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIM_CR1 register are not equal to 00. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the DIR direction bit in the TIM_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

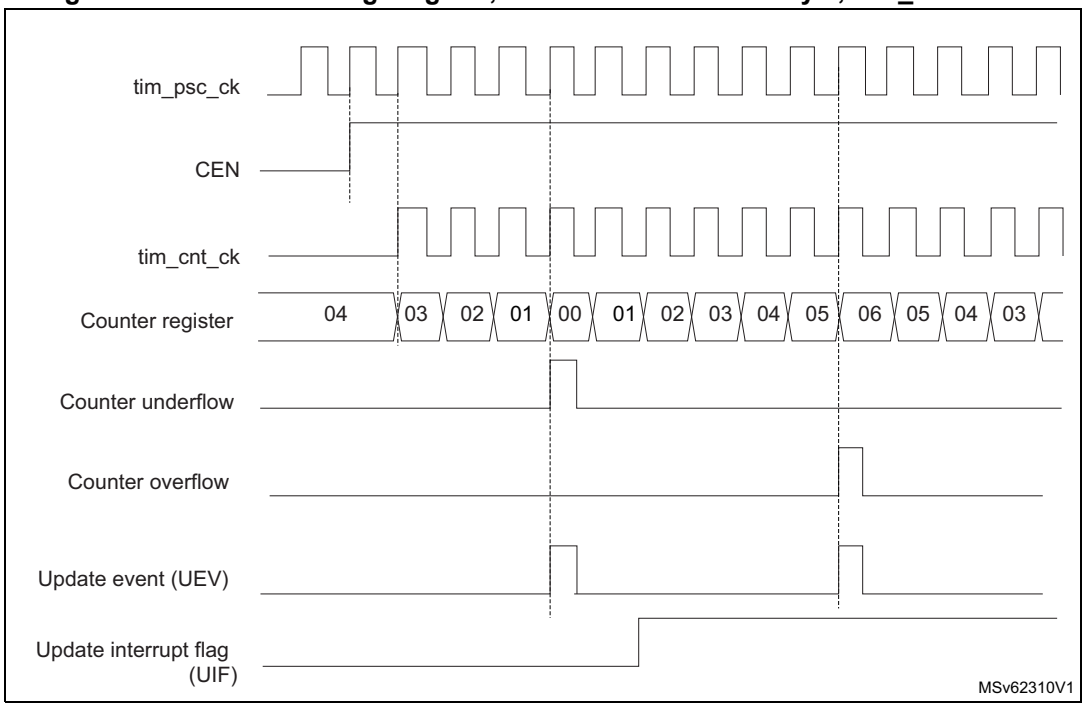
In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register)
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 528. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 34.6: TIM1/TIM8/TIM20 registers*).

Figure 529. Counter timing diagram, internal clock divided by 2

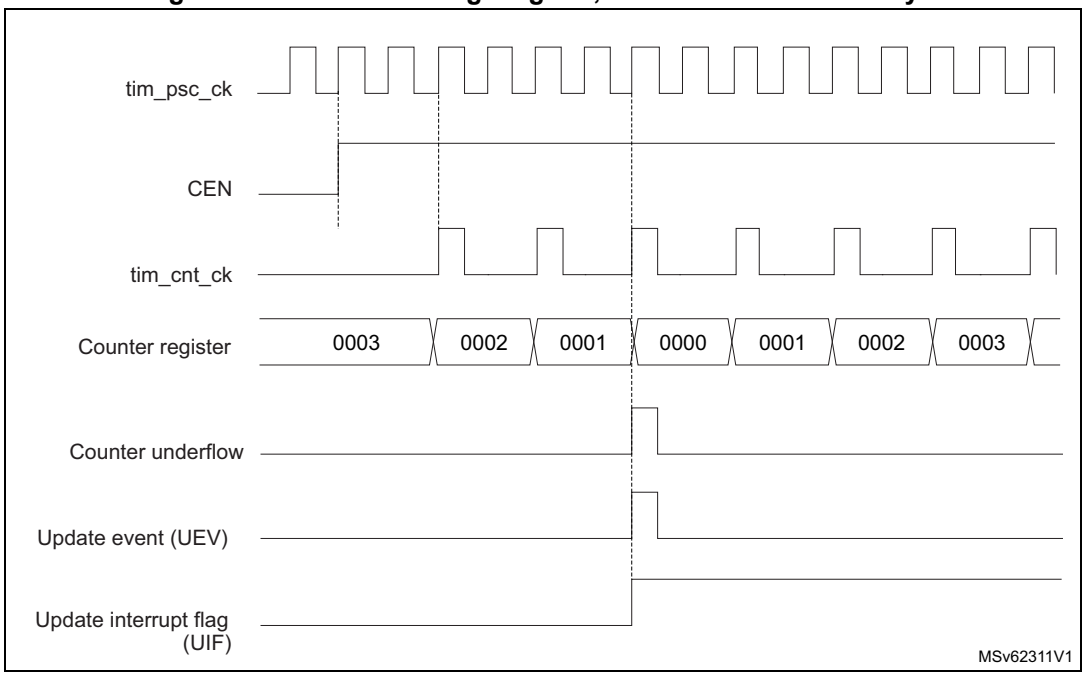


Figure 530. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36

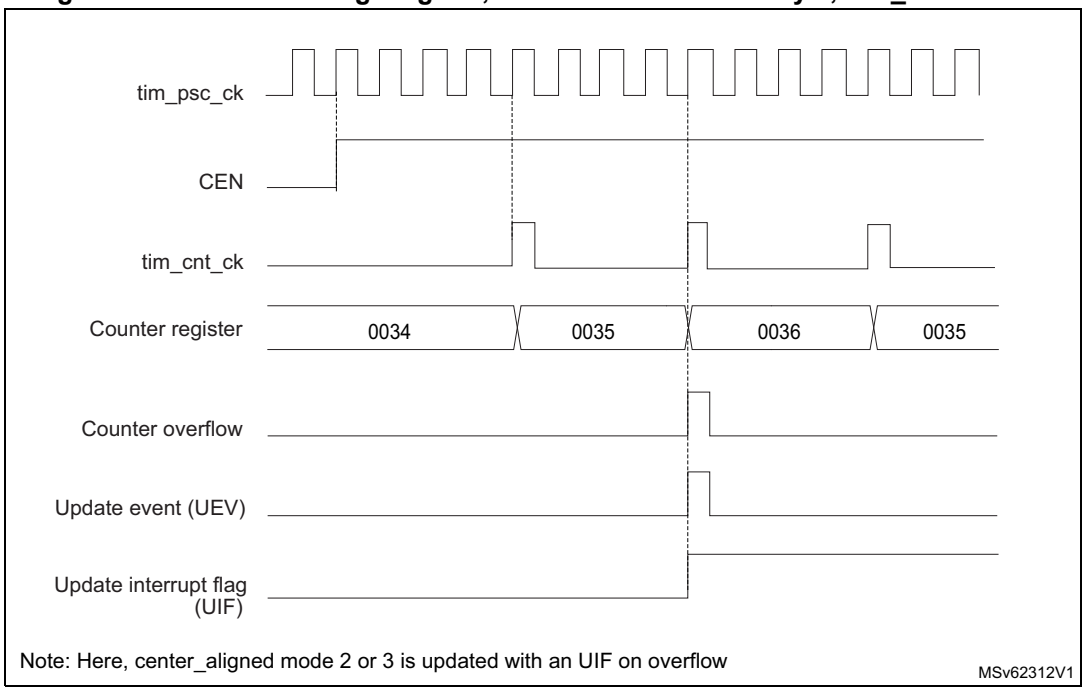


Figure 531. Counter timing diagram, internal clock divided by N

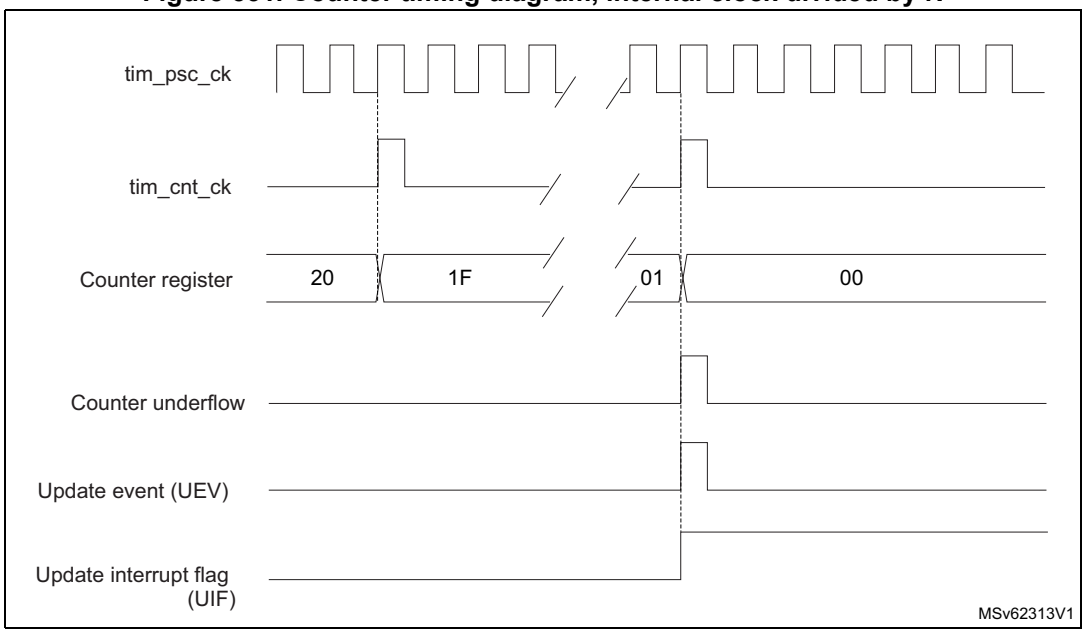


Figure 532. Counter timing diagram, update event with ARPE = 1 (counter underflow)

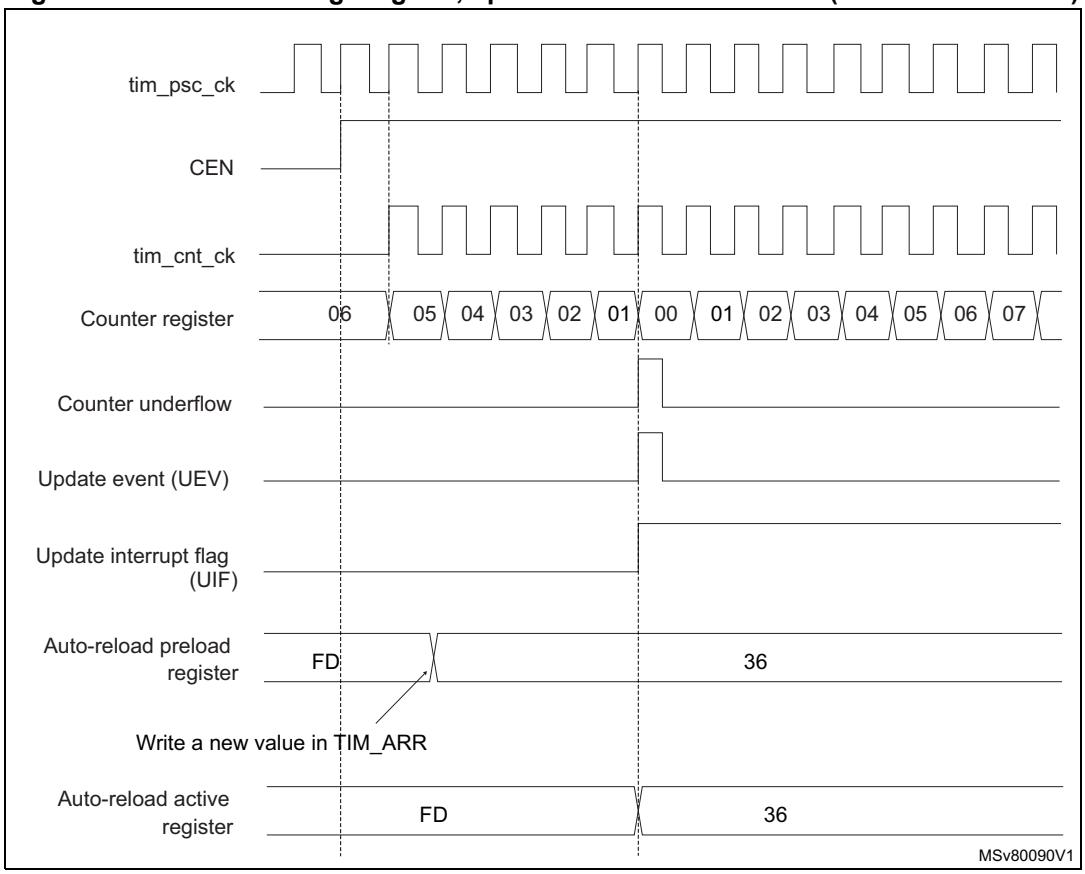
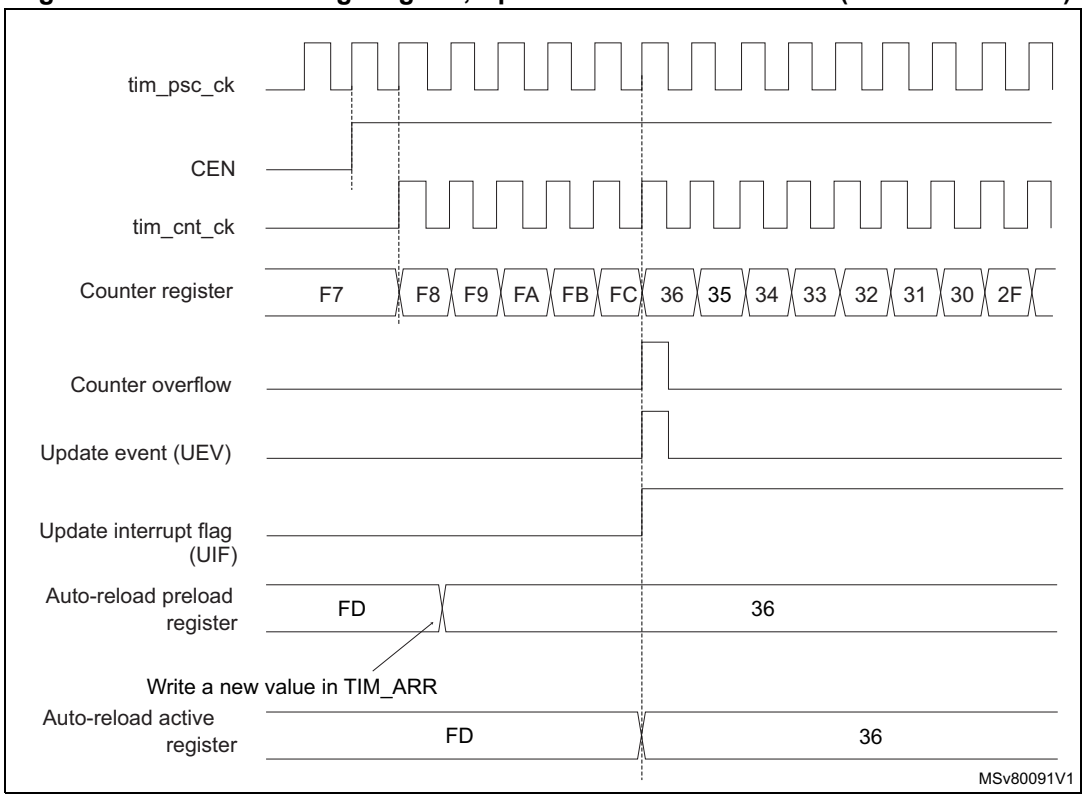


Figure 533. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



34.3.5 Repetition counter

Section 34.3.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIM_ARR autoreload register, TIM_PSC prescaler register, but also TIM_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIM_RCR repetition counter register.

The repetition counter is decremented:

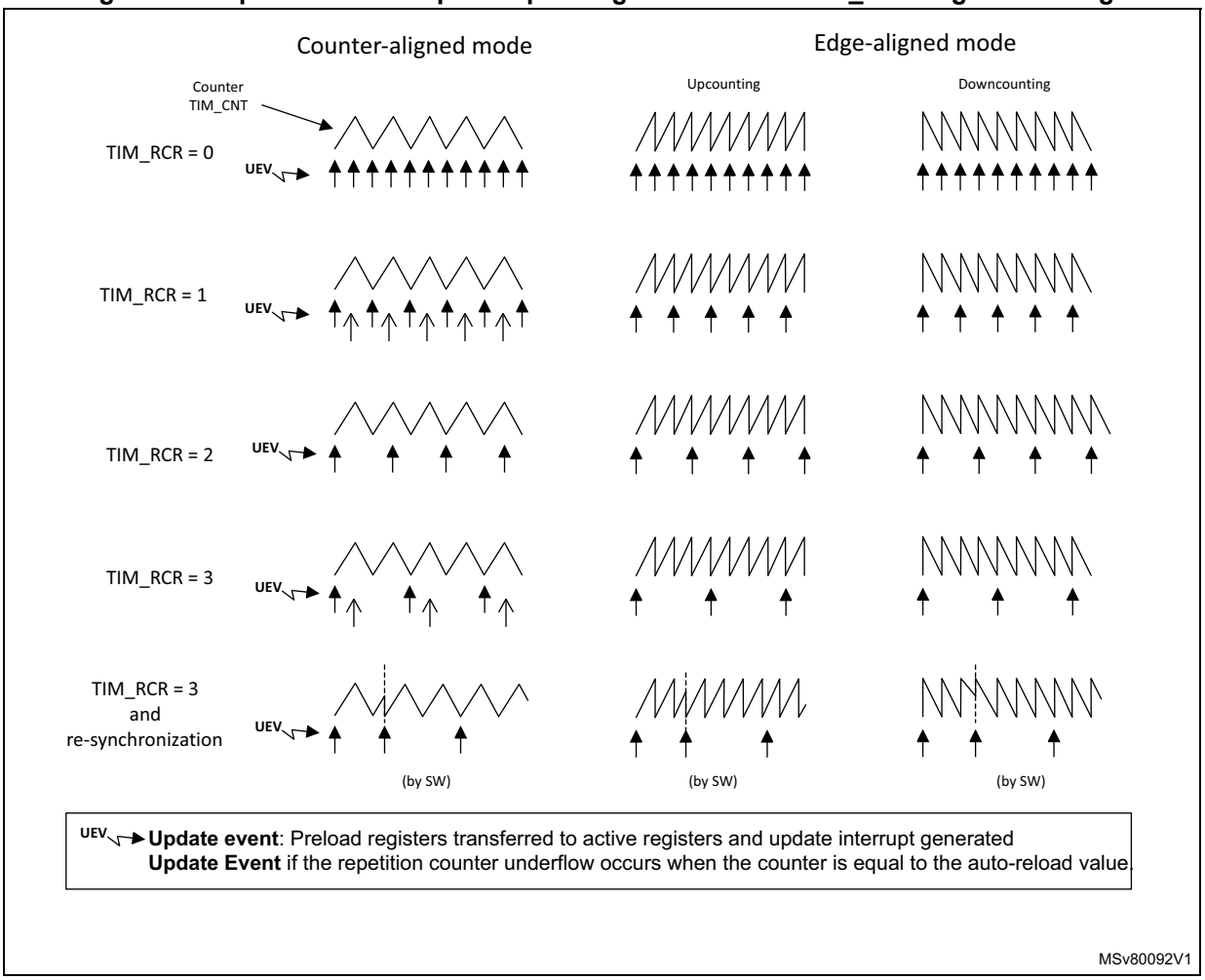
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIM_RCR register value (refer to *Figure 534*). When the update event is generated by software (by setting the UG bit in TIM_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIM_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 534. Update rate examples depending on mode and TIM_RCR register settings



34.3.6 External trigger input

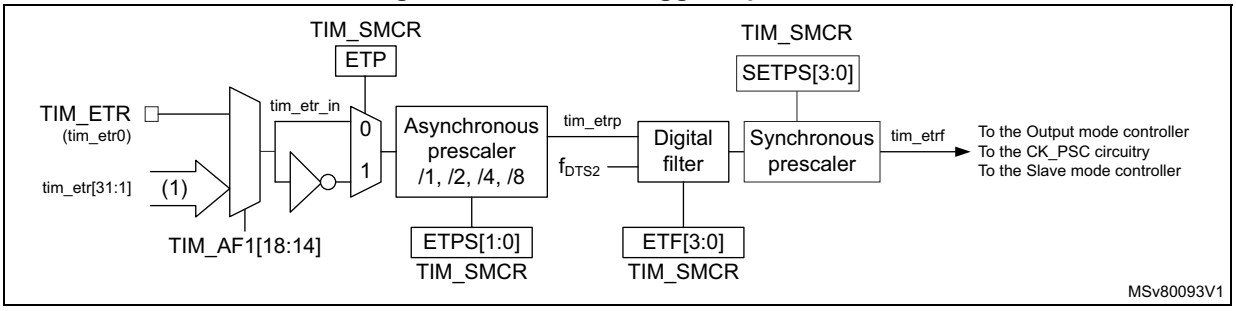
The timer features an external trigger input tim_etr_in. It can be used as:

- external clock (external clock mode 2, see *Section 34.3.7*)
- trigger for the slave mode (see *Section 34.3.35*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 34.3.9*)

Figure 535 below describes the tim_etr_in input conditioning. The input polarity is defined with the ETP bit in TIM_SMCR register. The trigger frequency can be decreased with the prescalers programmed by the ETPS[1:0] and SETPS[3:0] bitfields and digitally filtered as per the ETF[3:0] bitfield. The resulting signal (tim_etrf) is available for three purposes: as an

external clock, to condition the output (typically to reset a PWM output for a current limitation), and as a trigger for the Slave mode controller.

Figure 535. External trigger input block



The external trigger input block features 2 prescalers with different functionalities:

- An asynchronous prescaler (division by 1, 2, 4 or 8), able to process tim_etr_in signals with frequencies above tim_ker_ck
- A linear synchronous prescaler (division by 1,2,3,..15), only able to operate with tim_etrp signal frequencies below tim_ker_ck/3

Note: When tim_etrf is used as PWM reset input for cycle-by-cycle current regulation (see Section 34.3.23), SETPS[3:0] must be set to 0 (no prescaling).

Note: The SETPS[3:0] bitfield is always preloaded: a new value will become active only once an update event has occured.

The tim_etr_in input comes from multiple sources: input pins (default configuration), or internal sources. The selection is done with the ETRSEL[4:0] bitfield in the TIM_AF1 register.

Refer to *Section 34.3.2: TIM1/TIM8/TIM20 pins and internal signals* for the list of sources connected to the etr_in input in the product.

34.3.7 Clock selection

The counter clock can be provided by the following clock sources:

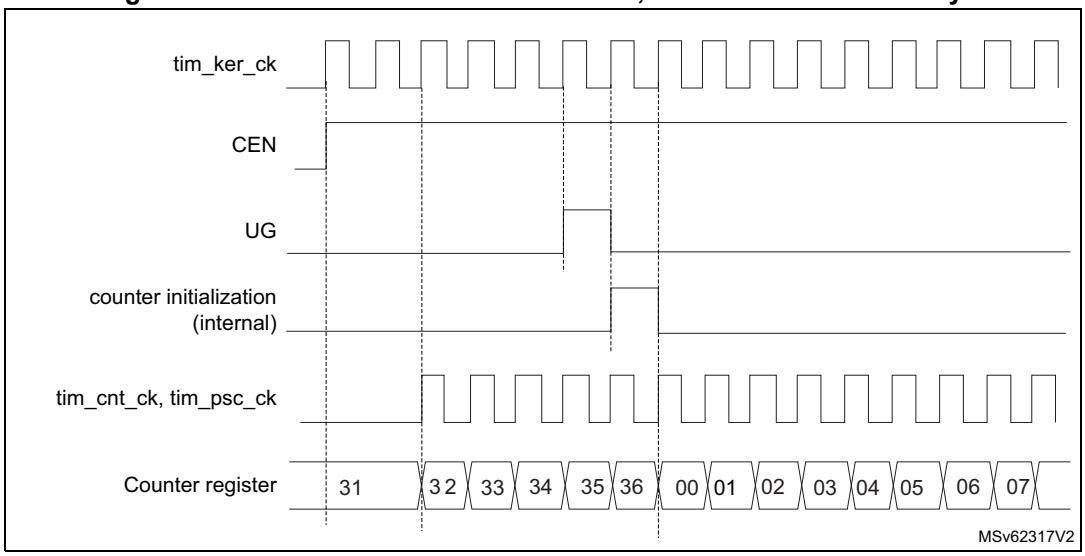
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2)
- External clock mode2: external trigger input (tim_etr_in)
- Encoder mode

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000), then the CEN, DIR (in the TIM_CR1 register) and UG bits (in the TIM_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 536 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

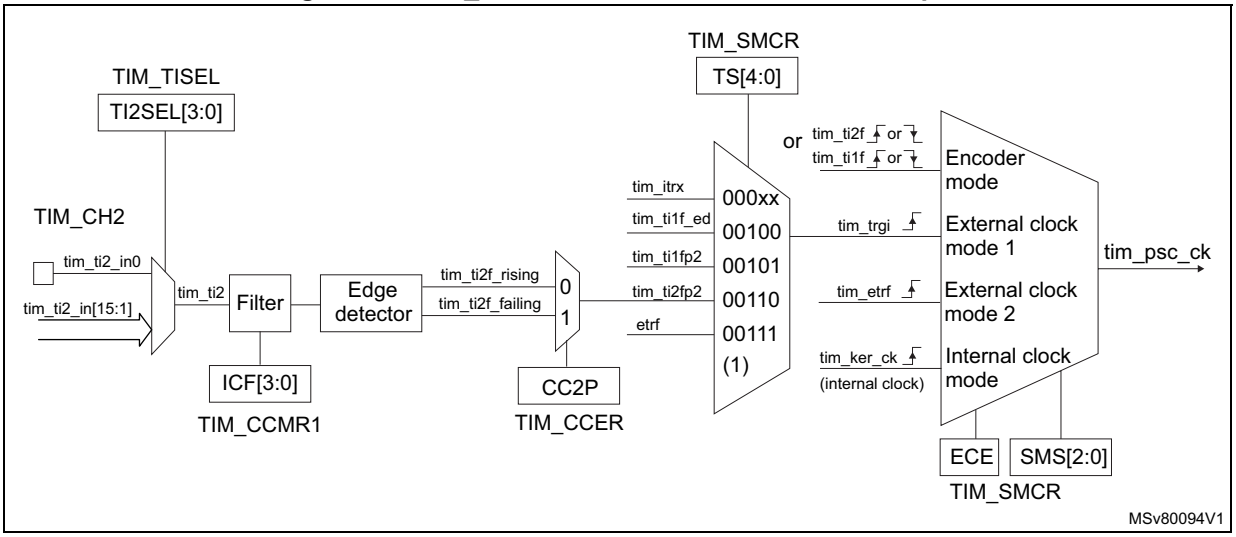
Figure 536. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 00111 in the TIM_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 537. tim_ti2 external clock connection example



1. Codes ranging from 01000 to 11111 are reserved.

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

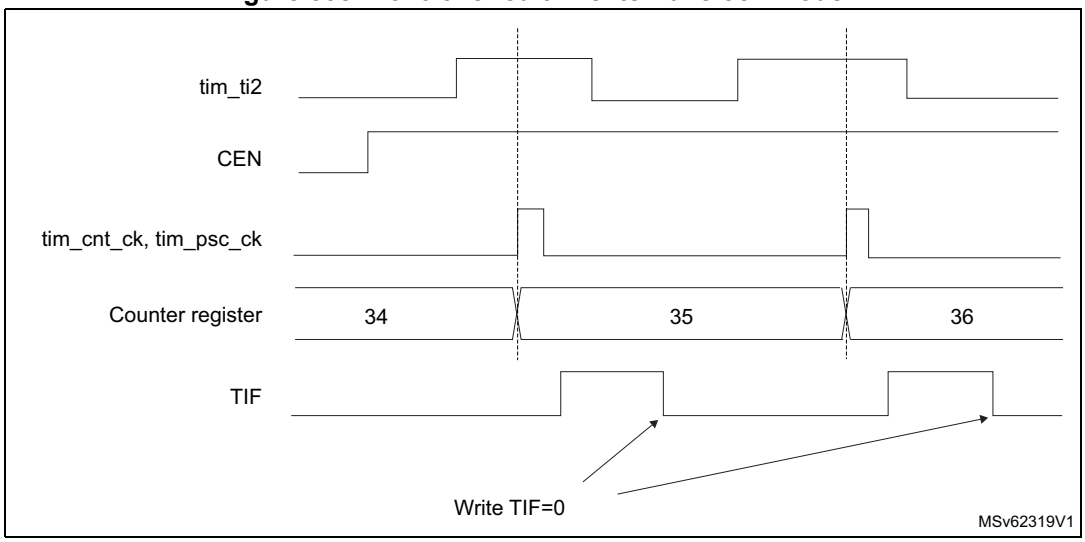
1. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIM_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIM_CCMR1 register (if no filter is needed, keep IC2F = 0000).
3. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIM_SMCR register.
5. Select tim_ti2 as the trigger input source by writing TS = 00110 in the TIM_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 538. Control circuit in external clock mode 1



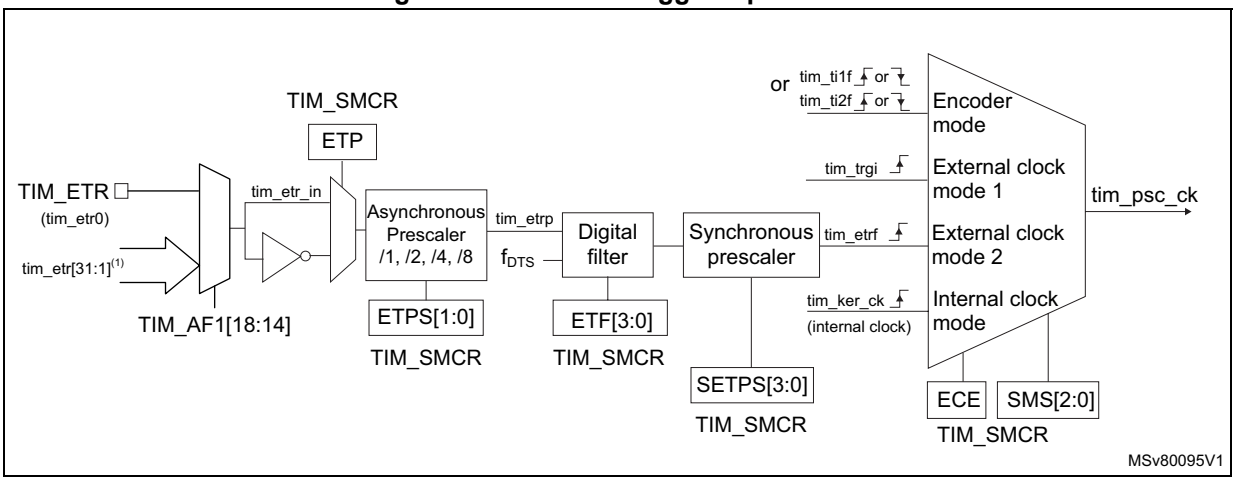
External clock source mode 2

This mode is selected by writing ECE = 1 in the TIM_SMCR register.

The counter counts at each rising or falling edge on the external trigger input tim_etr_in.

The *Figure 539* gives an overview of the external trigger input block.

Figure 539. External trigger input block



1. Refer to *Section 34.3.2: TIM1/TIM8/TIM20 pins and internal signals*.

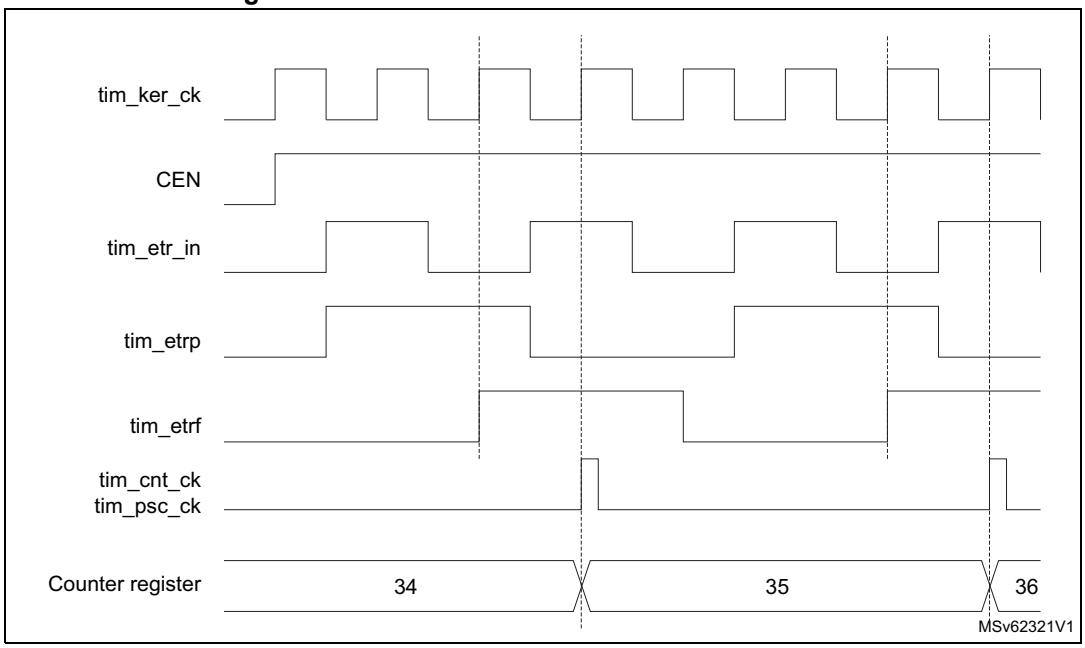
For example, to configure the upcounter to count each 2 rising edges on tim_etr_in, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIM_SMCR register.
2. Set the prescaler by writing ETPS[1:0] = 01 and SETPS[3:0] = 0 in the TIM_SMCR register
3. Select rising edge detection on the tim_etr_in input by writing ETP = 0 in the TIM_SMCR register
4. Enable external clock mode 2 by writing ECE = 1 in the TIM_SMCR register.
5. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

The counter counts once each 2 tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of tim_ker_ck frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 540. Control circuit in external clock mode 2



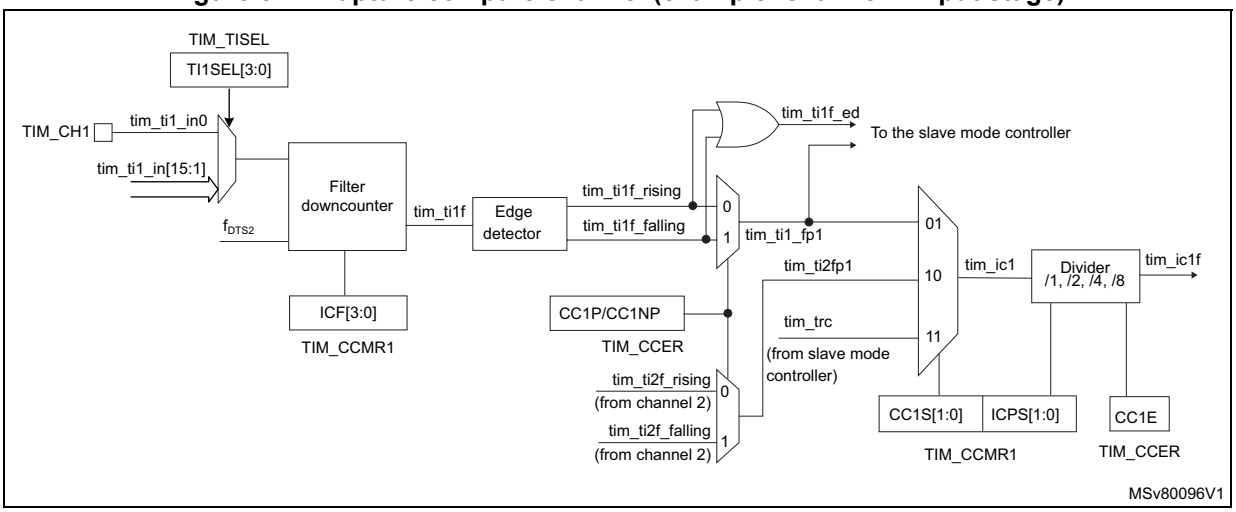
34.3.8 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 541 to *Figure 544* give an overview of one capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 541. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 542. Capture/compare channel 1 main circuit

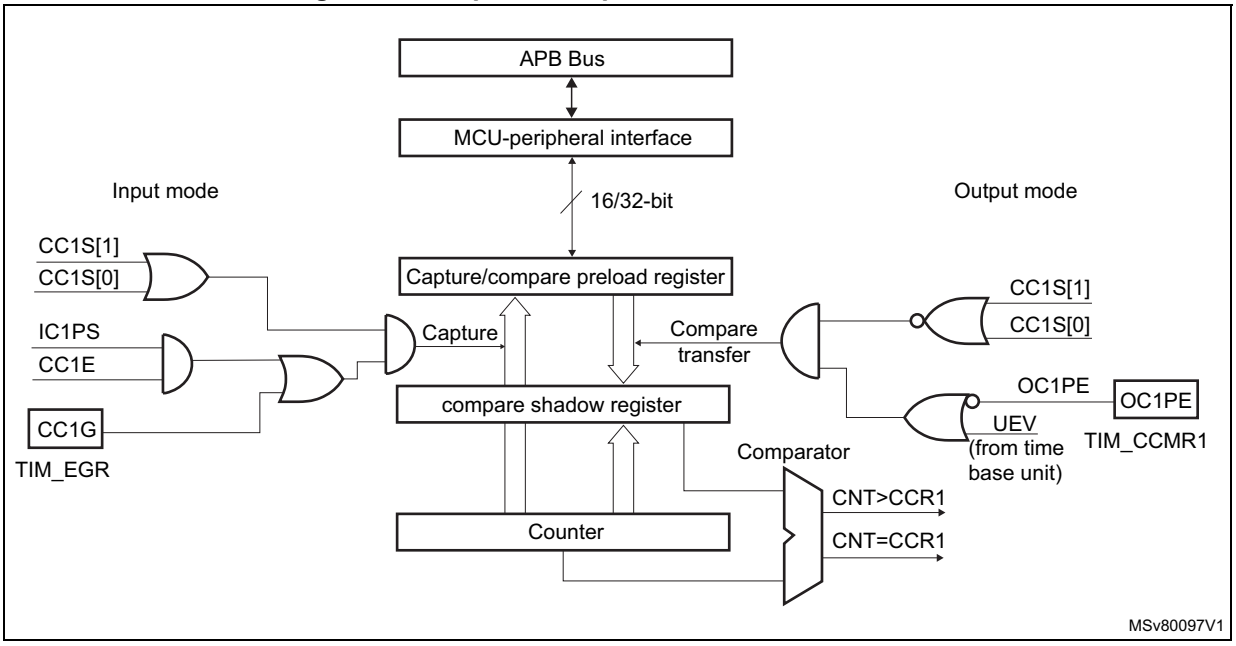
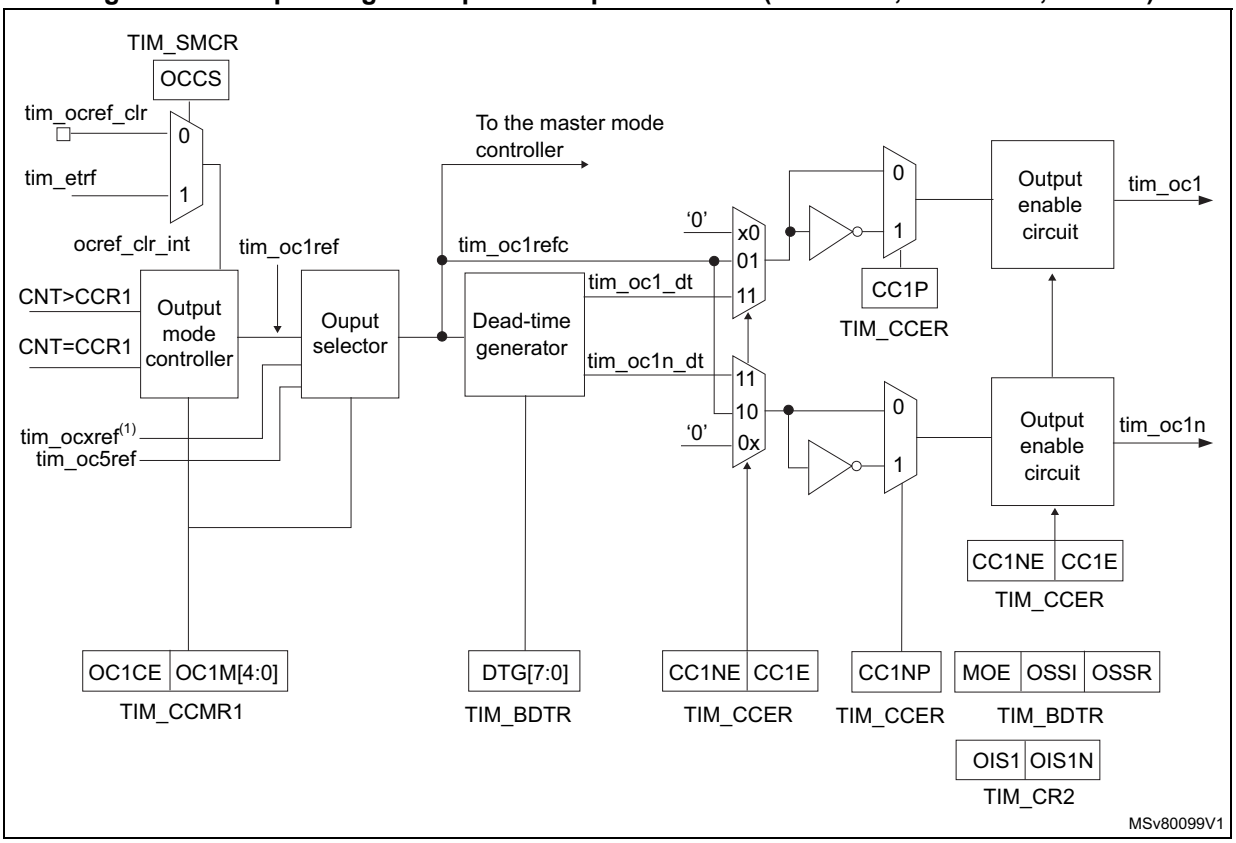
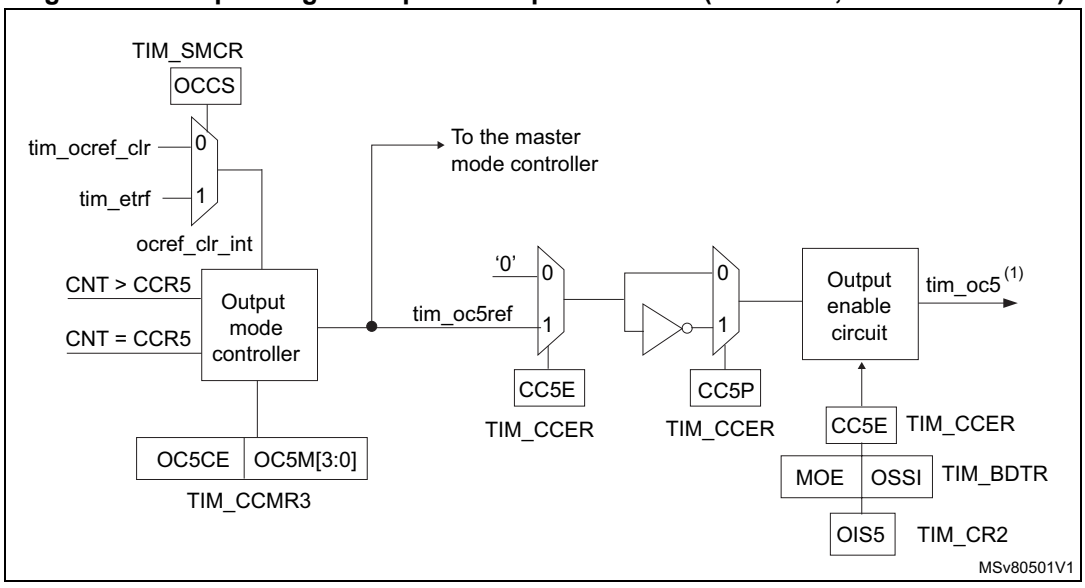


Figure 543. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4)



1. tim_ocxref, where x is the rank of the complementary channel

Figure 544. Output stage of capture/compare channel (channel 5, idem ch. 6 and 7)



1. Not available externally.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

34.3.9 Input capture mode

In Input capture mode, the capture/compare registers (TIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIM_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIM_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIM_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIM_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

- Select the active input: TIM_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIM_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input, and the TIM_CCR1 register becomes read-only.
- Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIM_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS2} frequency). Then write IC1F bits to 0011 in the TIM_CCMR1 register.
- Select the edge of the active transition on the tim_ti1 channel by writing CC1P and CC1NP bits to 0 in the TIM_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIM_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIM_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIM_DIER register, and/or the DMA request by setting the CC1DE bit in the TIM_DIER register.

When an input capture occurs:

- The TIM_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIM_EGR register.

34.3.10 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIM_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulsewidth (interval between two consecutive rising and falling edges.

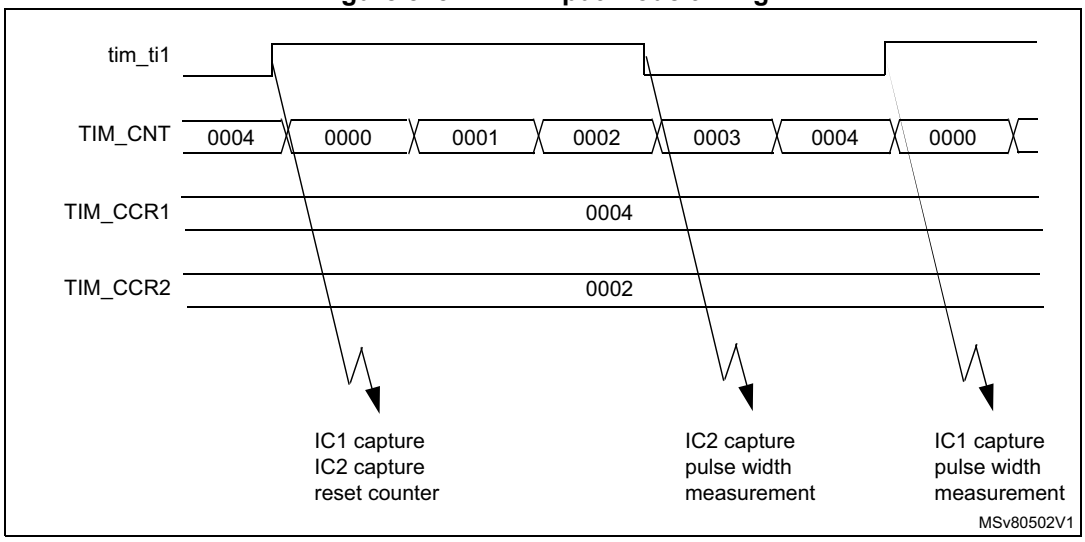
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tixfp1 input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two tim_tixfp signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulsewidth of a PWM signal applied on tim_ti1 can be measured using the following procedure:

- Select the active input for TIM_CCR1: write the CC1S bits to 01 in the TIM_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp1 (used both for capture in TIM_CCR1 and counter clear): write the CC1P and CC1NP bits to 0 (active on rising edge).
- Select the active input for TIM_CCR2: write the CC2S bits to 10 in the TIM_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp2 (used for capture in TIM_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP = 10 (active on falling edge).
- Select the valid trigger input: write the TS bits to 00101 in the TIM_SMCR register (tim_ti1fp1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIM_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to 1 in the TIM_CCER register.

Figure 545. PWM input mode timing



34.3.11 Forced output mode

In output mode (CCxS bits = 00 in the TIM_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, user just needs to write 00101 in the OCxM bits in the corresponding TIM_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 00100 in the TIM_CCMRx register.

Anyway, the comparison between the TIM_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

34.3.12 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while channel 5 to 7 are only available inside the microcontroller (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIM_CCMRx register) and the output polarity (CCxP bit in the TIM_CCER register). The output pin can keep its level (OCxM = 00000), be set active (OCxM = 00001), be set inactive (OCxM = 00010) or can toggle (OCxM = 00011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIM_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIM_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIM_DIER register, CCDS bit in the TIM_CR2 register for the DMA request selection).

The TIM_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIM_CCMRx register.

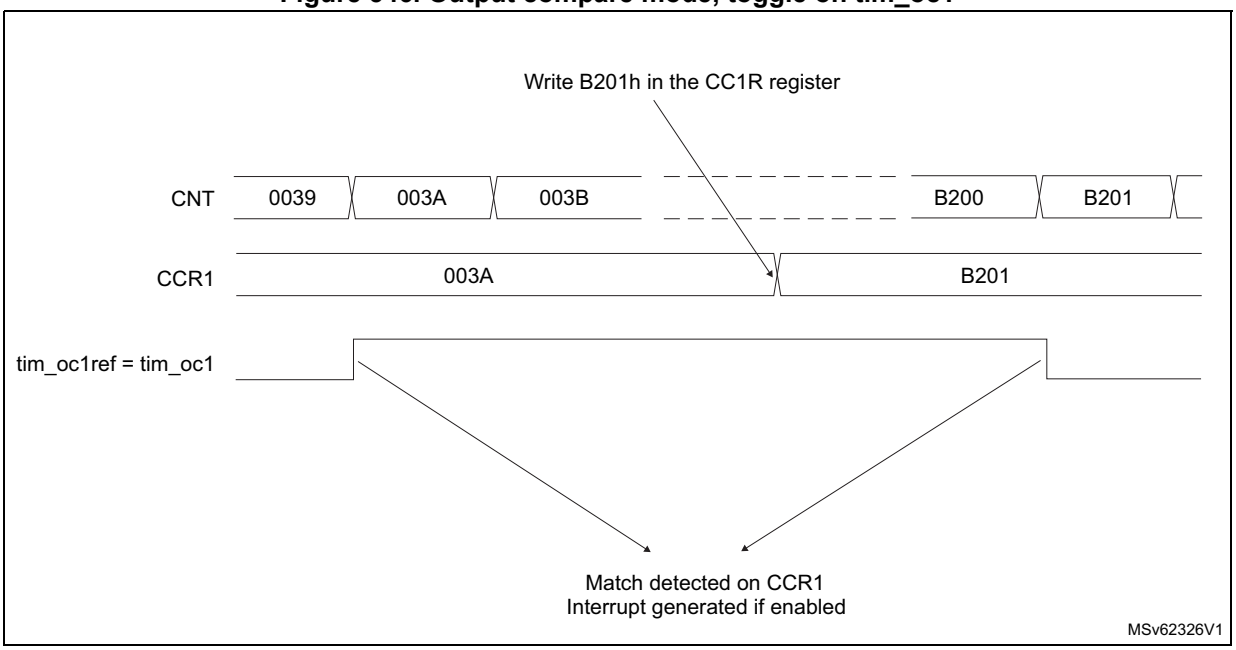
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIM_ARR and TIM_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 00011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIM_CR1 register.

The TIM_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIM_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 546*.

Figure 546. Output compare mode, toggle on tim_oc1



34.3.13 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIM_ARR register and a duty cycle determined by the value of the TIM_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 00110 (PWM mode 1) or 00111 (PWM mode 2) in the OCxM bits in the TIM_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIM_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIM_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIM_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIM_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (TIM_CCER and TIM_BDTR registers). Refer to the TIM_CCER register description for more details.

In PWM mode (1 or 2), TIM_CNT and TIM_CCRx are always compared to determine whether TIM_CCRx ≤ TIM_CNT or TIM_CNT ≤ TIM_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIM_CR1 register.

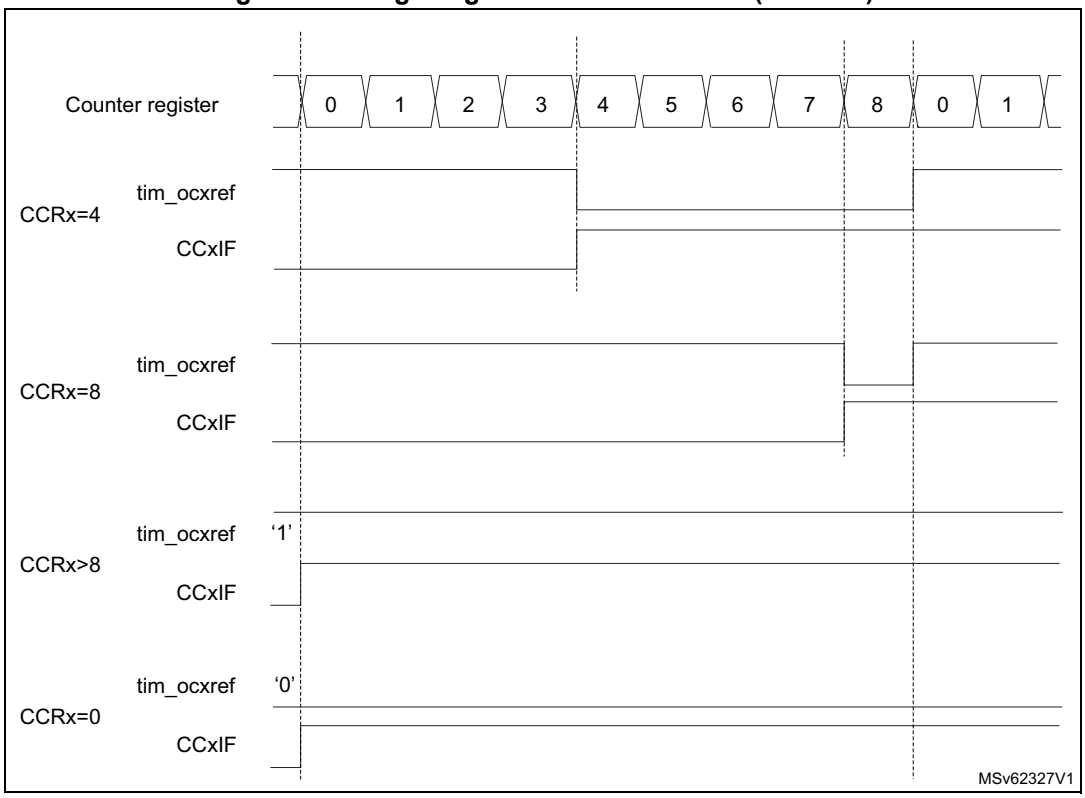
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIM_CR1 register is low. Refer to *Upcounting mode*.

  In the following example, the mode is PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIM_CNT < TIM_CCRx else it becomes low. If the compare value in TIM_CCRx is greater than the autoreload value (in TIM_ARR) then tim_ocxref is held at 1. If the compare value is zero then tim_ocxref is held at 0. *Figure 547* shows some edge-aligned PWM waveforms in an example where TIM_ARR = 8.

Figure 547. Edge-aligned PWM waveforms (ARR = 8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIM_CR1 register is high. Refer to the *Downcounting mode*

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIM_CNT > TIM_CCRx else it becomes high. If the compare value in TIM_CCRx is greater than the autoreload value in TIM_ARR, then tim_ocxref is held at 1. 0% PWM is not possible in this mode.

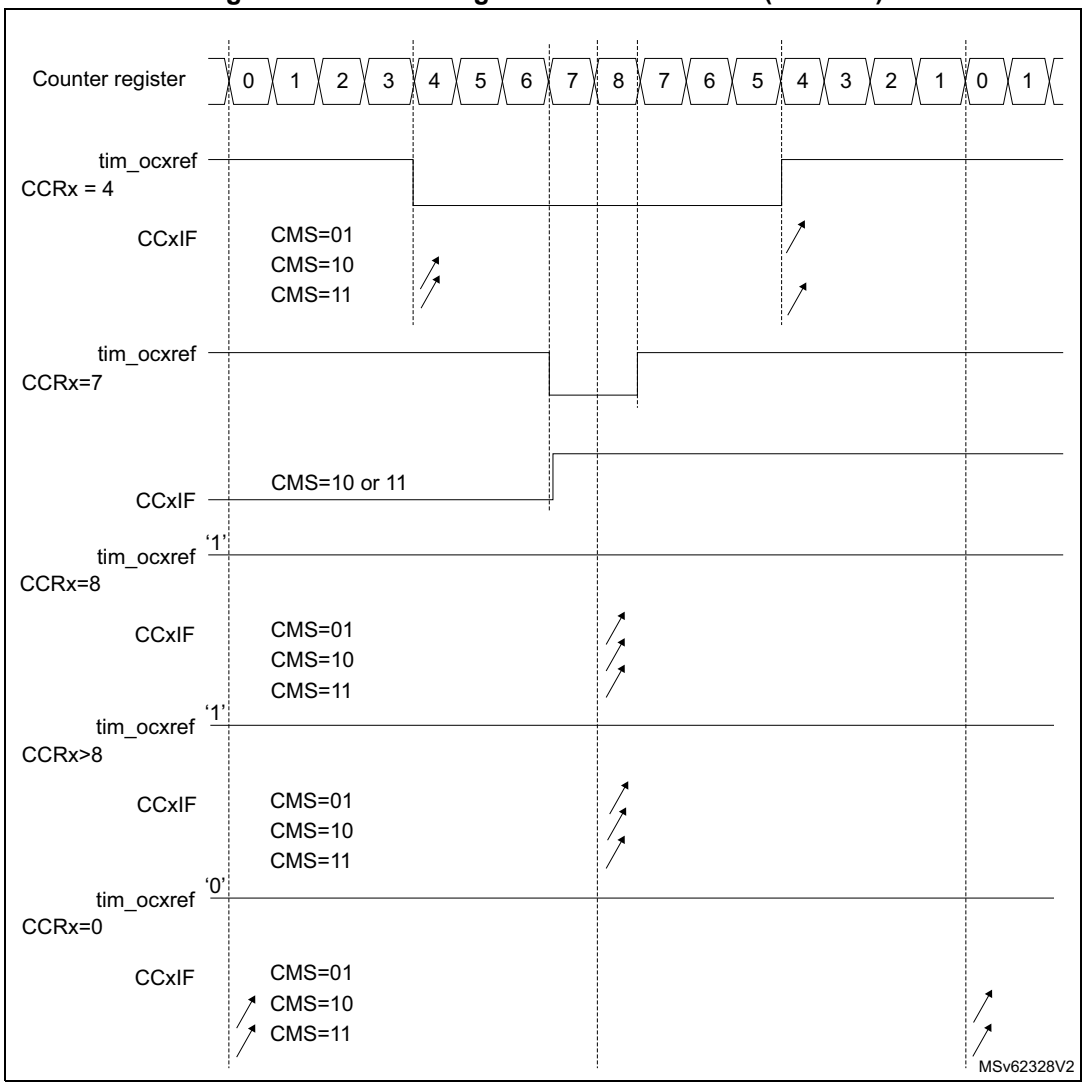
PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIM_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIM_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting)*.

Figure 548 shows some center-aligned PWM waveforms in an example where:

- TIM_ARR = 8
- PWM mode is the PWM mode 1
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIM_CR1 register.

Figure 548. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

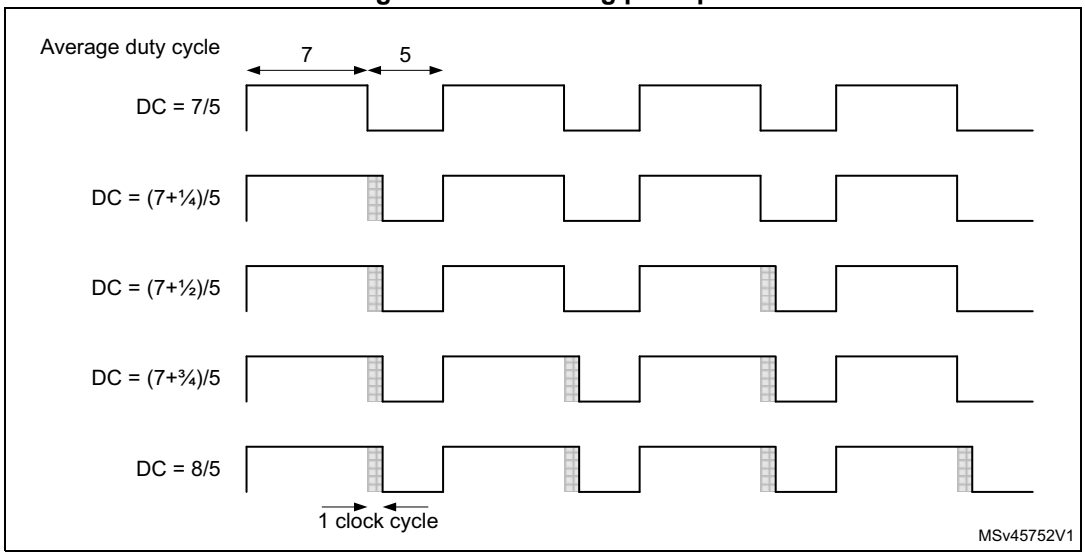
- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIM_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIM_CNT > TIM_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIM_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIM_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIM_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 549* presents the dithering principle applied to four consecutive PWM cycles.

Figure 549. Dithering principle



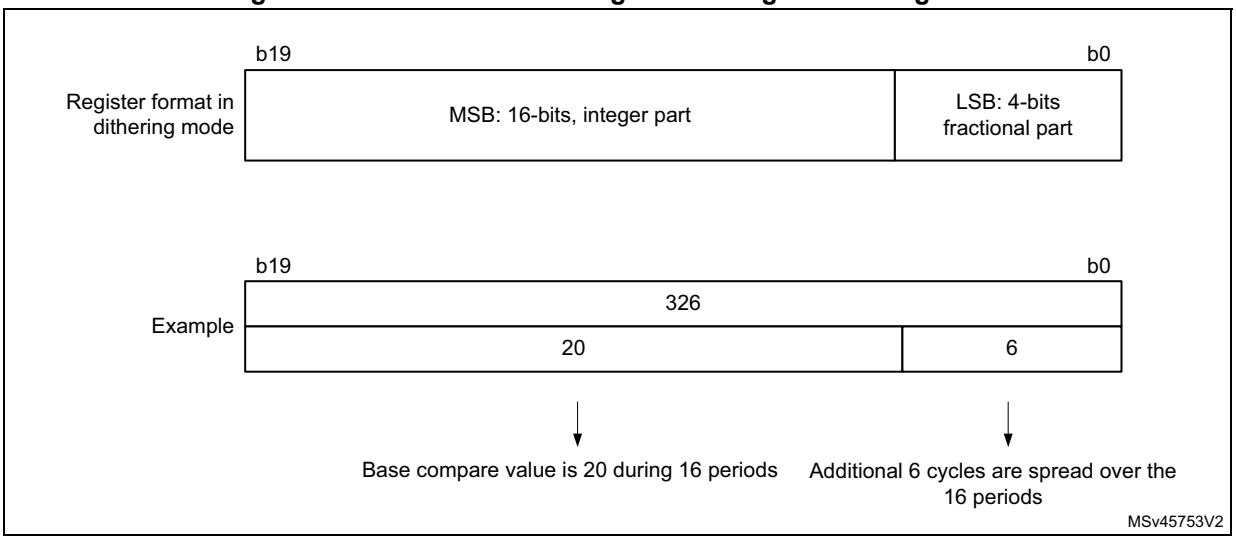
When the dithering mode is enabled, the register coding is changed as follows (see *Figure 550* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:

1. *CEN and ARPE bits must be reset.*
2. *The DITHEN bit must be reset.*
3. *The CCIF flags must be cleared.*
4. *The CEN bit can be set (eventually with ARPE = 1).*

Figure 550. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

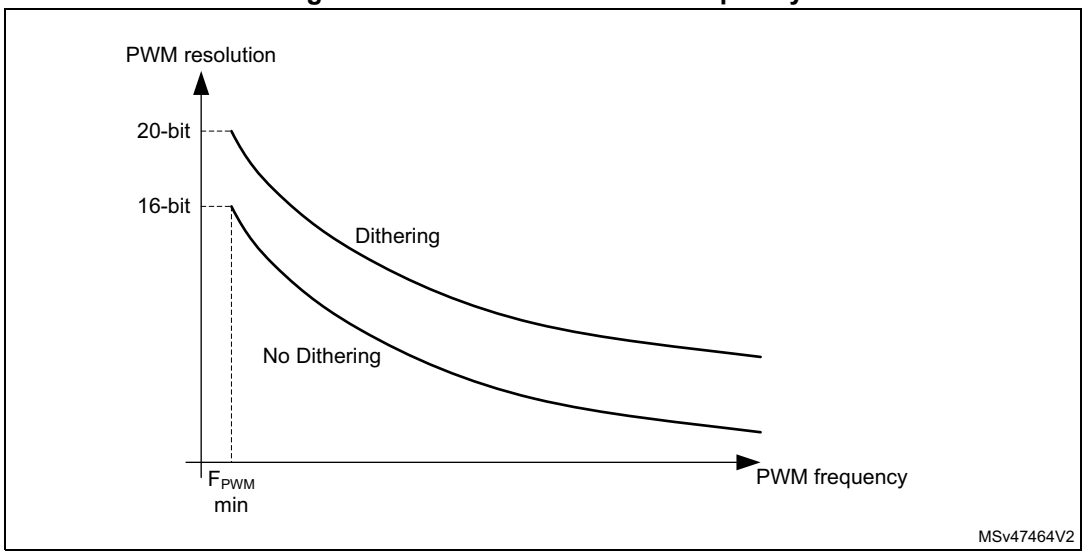
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIM_ARR and TIM_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

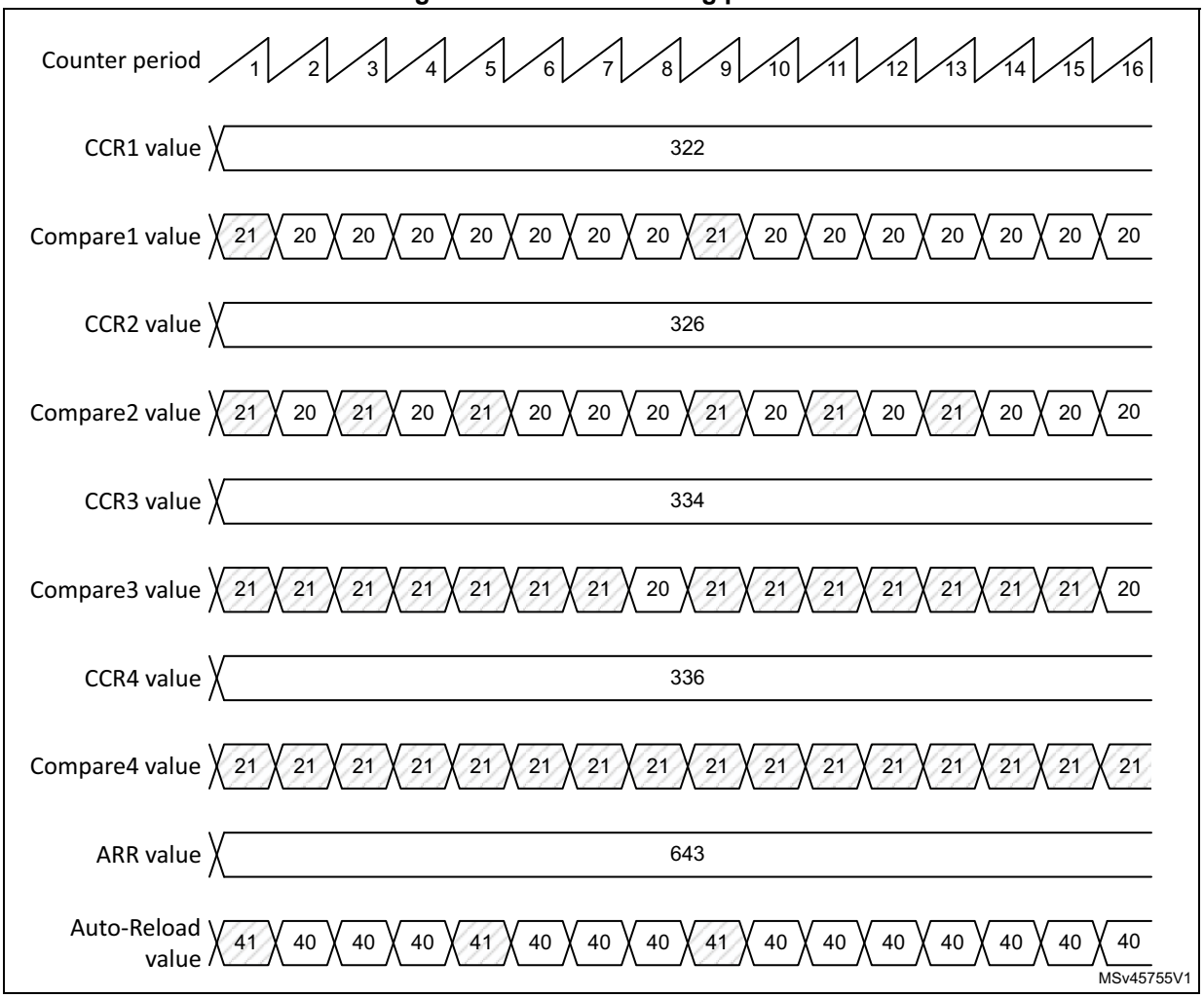
As shown on *Figure 551*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 551. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 552*.

Figure 552. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 379*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 379. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 379. CCR and ARR register change dithering pattern (continued)

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIM_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down counting phases as shown in *Figure 553*.

Figure 553. Dithering effect on duty cycle in center-aligned PWM mode

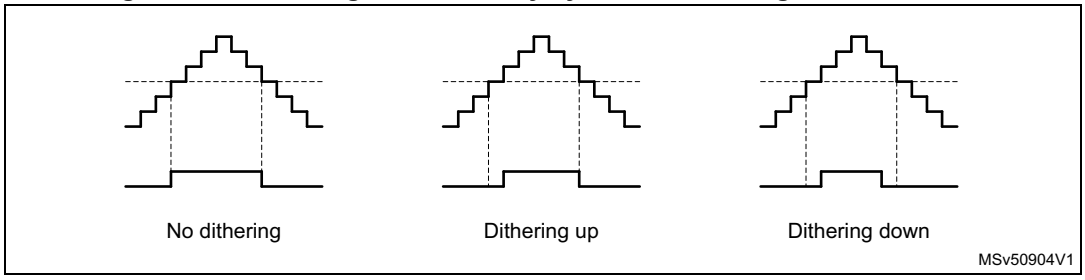


Table 380 shows how the dithering pattern is added in center-aligned PWM mode.

Table 380. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 380. CCR register change dithering pattern in center-aligned PWM mode (continued)

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

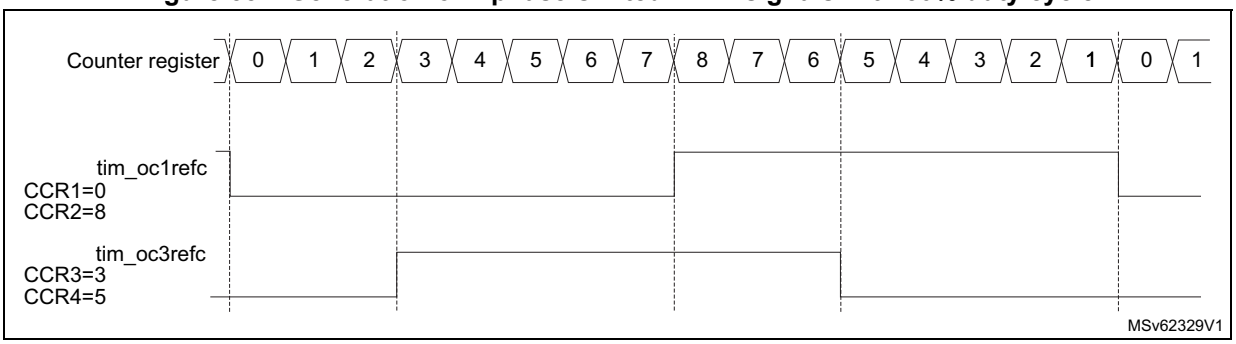
34.3.14 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and the phase-shift are determined by a pair of TIM_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle, for instance:

- tim_oc1refc is controlled by TIM_CCR1 and TIM_CCR2
- tim_oc3refc is controlled by TIM_CCR3 and TIM_CCR4
- ...

Figure 554 represents an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 554. Generation of 2 phase-shifted PWM signals with 50% duty cycle



Multiple asymmetric PWM modes are available to combine different pairs of TIM_CCRx registers and generate the required waveform, as listed in *Table 381*. The selection is done

with the OCxM[4:0] bitfield in TIM_CCMRx registers. For each pair of TIM_CCRx registers, two options are given (e.g. Asymmetric PWM mode 3 and 4), one being in PWM mode 1 (channel active as long as the counter is below the TIM_CCRx value), the other one in PWM mode 2 (channel active when the counter is above the TIM_CCRx value).

For the asymmetric PWM modes 1 to 6, it is mandatory to have the same PWM mode for both channels. For instance, it is mandatory to have OC1M[4:0] = OC2M[4:0] = 00110 for the asymmetric PWM mode 1 on oc1refc.

For the asymmetric PWM modes 7 to 10, it is mandatory to have opposite PWM mode for both channels (one being in PWM mode 1 and the other in PWM mode 2). Refer to the *Table 381* for configuration.

Note: The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bits are not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the regular tim_oc2ref signal on channel 2 or a tim_oc2refc signal resulting from asymmetric PWM mode 1.

Table 381. Asymmetric PWM modes

<table>
<tr><th>Output compare mode</th><th>OCxM[4:0]</th><th>oc1refc</th><th>oc2refc</th><th>oc3refc</th><th>oc4refc</th><th>oc5refc</th><th>oc6refc</th><th>oc7refc</th><th>PWM Mode (1)</th></tr>
<tr><td>Asymmetric PWM mode 1</td><td>01110</td><td colspan="2" rowspan="2">oc1ref (up-counting)
oc2ref (down-counting)</td><td colspan="2" rowspan="2">oc3ref (up-counting)
oc4ref (down-counting)</td><td colspan="2" rowspan="2">oc5ref (up-counting)
oc6ref (down-counting)</td><td rowspan="2">-</td><td>1</td></tr>
<tr><td>Asymmetric PWM mode 2</td><td>01111</td><td>2</td></tr>
<tr><td>Asymmetric PWM mode 3</td><td>10010</td><td>oc1ref (up-counting)
oc5ref (down-counting)</td><td>oc2ref (up-counting)
oc5ref (down-counting)</td><td>oc3ref (up-counting)
oc5ref (down-counting)</td><td>oc4ref (up-counting)
oc5ref (down-counting)</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">-</td><td>1</td></tr>
<tr><td>Asymmetric PWM mode 4</td><td>10011</td><td>2</td></tr>
<tr><td>Asymmetric PWM mode 5</td><td>10100</td><td>oc1ref (up-counting)
oc6ref (down-counting)</td><td>oc2ref (up-counting)
oc6ref (down-counting)</td><td>oc3ref (up-counting)
oc6ref (down-counting)</td><td>oc4ref (up-counting)
oc6ref (down-counting)</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">-</td><td>1</td></tr>
<tr><td>Asymmetric PWM mode 6</td><td>10101</td><td>2</td></tr>
</table>

Table 381. Asymmetric PWM modes

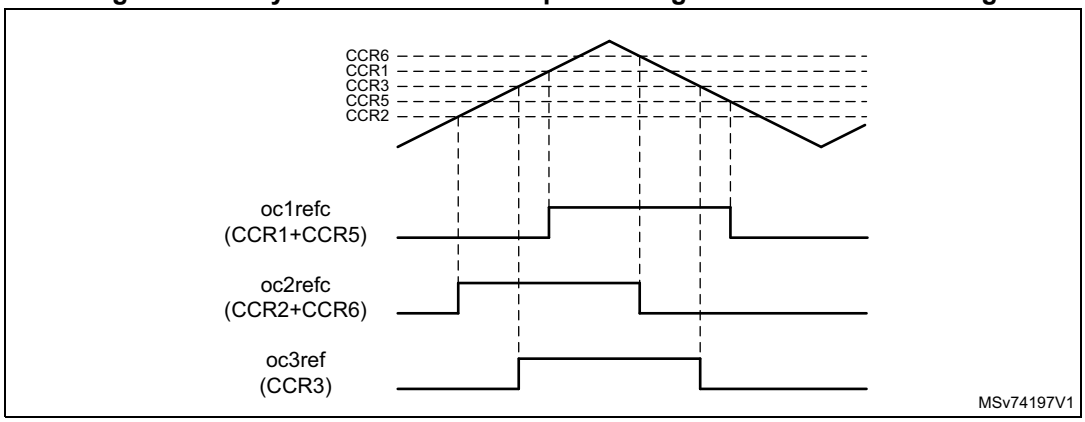
Output compare mode	OCxM[4:0]	oc1refc	oc2refc	oc3refc	oc4refc	oc5refc	oc6refc	oc7refc	PWM Mode (1)
Asymmetric PWM mode 7	10110	oc1ref OR oc5ref (up-counting) oc1refc=1 (down-counting)	oc2ref OR oc6ref (up-counting) oc2refc=1 (down-counting)	oc3ref OR oc4ref (up-counting) oc3refc=1 (down-counting)	oc4ref OR oc7ref (up-counting) oc4refc=1 (down-counting)	-	-	-	1(2)
Asymmetric PWM mode 8	10111	oc1ref AND oc5ref (up-counting) oc1refc=0 (down-counting)	oc2ref AND oc6ref (up-counting) oc2refc=0 (down-counting)	oc3ref AND oc4ref (up-counting) oc3refc=0 (down-counting)	oc4ref AND oc7ref (up-counting) oc4refc=0 (down-counting)				2(3)
Asymmetric PWM mode 9	11000	oc1refc=1 (up-counting) oc1ref OR oc5ref (down-counting)	oc2refc=1 (up-counting) oc2ref OR oc6ref (down-counting)	oc3refc=1 (up-counting) oc3ref OR oc4ref (down-counting)	oc4refc=1 (up-counting) oc4ref OR oc7ref (down-counting)	-	-	-	1(2)
Asymmetric PWM mode 10	11001	oc1refc=0 (up-counting) oc1ref AND oc5ref (down-counting)	oc2refc=0 (up-counting) oc2ref AND oc6ref (down-counting)	oc3refc=0 (up-counting) oc3ref AND oc4ref (down-counting)	oc4refc=0 (up-counting) oc7ref AND oc4ref (down-counting)				2(3)

1. PWM mode 1: channel active as long as the counter is below the TIM_CCRx value). PWM mode 2: channel active when the counter is above the TIM_CCRx value. Unless explicitly mentioned, the PWM mode must be identical for the 2 channels used for the asymmetric mode
2. oc5ref, oc6ref, oc4ref and oc7ref must be programmed in PWM mode 2
3. oc5ref, oc6ref, oc4ref and oc7ref must be programmed in PWM mode 1

The *Figure 555* represents an example suitable for single-shunt current sensing techniques requiring phase-shifted PWMs, for 3-phase motor control. This is done using asymmetric PWM mode 3 and 5:

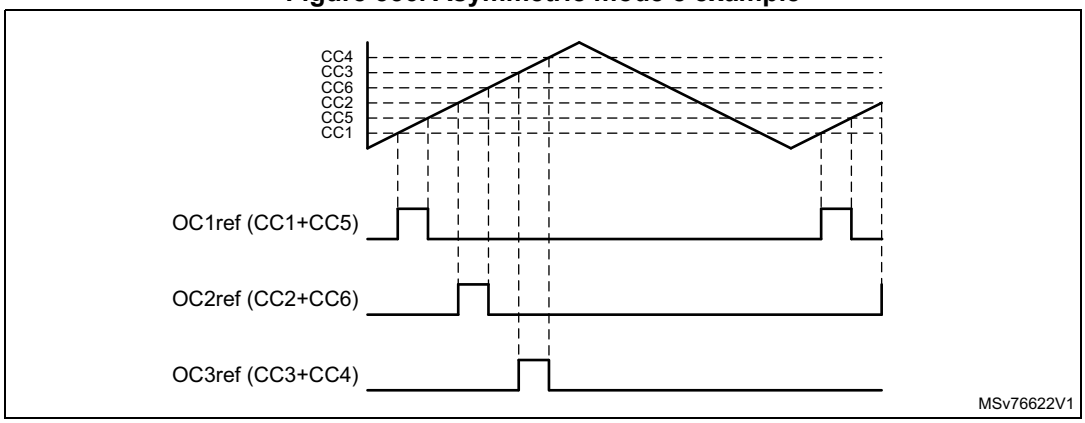
- Channel 1 is configured in asymmetric PWM mode 4
- Channel 2 is configured in asymmetric PWM mode 5
- Channel 3 is configured in regular PWM mode 2
- Compare 4 and 7 are available for ADC triggering purpose

Figure 555. Asymmetric mode example for single-shunt current sensing



The *Figure 556* represents an example of the waveforms that can be generated with asymmetric modes 7 to 10.

Figure 556. Asymmetric mode 8 example



34.3.15 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and delay are determined by the two TIM_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs, as summarized in the below:

Table 382. Combined PWM mode overview

Output compare mode	OCxM[4:0]	oc1refc	oc2refc	oc3refc	oc4refc	oc5refc	oc6refc	oc7refc	PWM mode(1)
Combined PWM mode 1	01100	oc1ref OR oc2ref		oc3ref OR oc4ref		oc5ref OR oc6ref		-	1
Combined PWM mode 2	01101	oc1ref AND oc2ref		oc3ref AND oc4ref		oc5ref AND oc6ref		-	2
Combined PWM mode 3	10000	oc1ref OR oc5ref	oc2ref OR oc6ref	oc3ref OR oc5ref	oc4ref OR oc6ref	-	-	-	1
Combined PWM mode 4	10001	oc1ref AND oc5ref	oc2ref AND oc6ref	oc3ref AND oc5ref	oc4ref AND oc6ref	-	-	-	2

1. PWM mode 1: channel active as long as the counter is below the TIM_CCRx value). PWM mode 2: channel active when the counter is above the TIM_CCRx value.

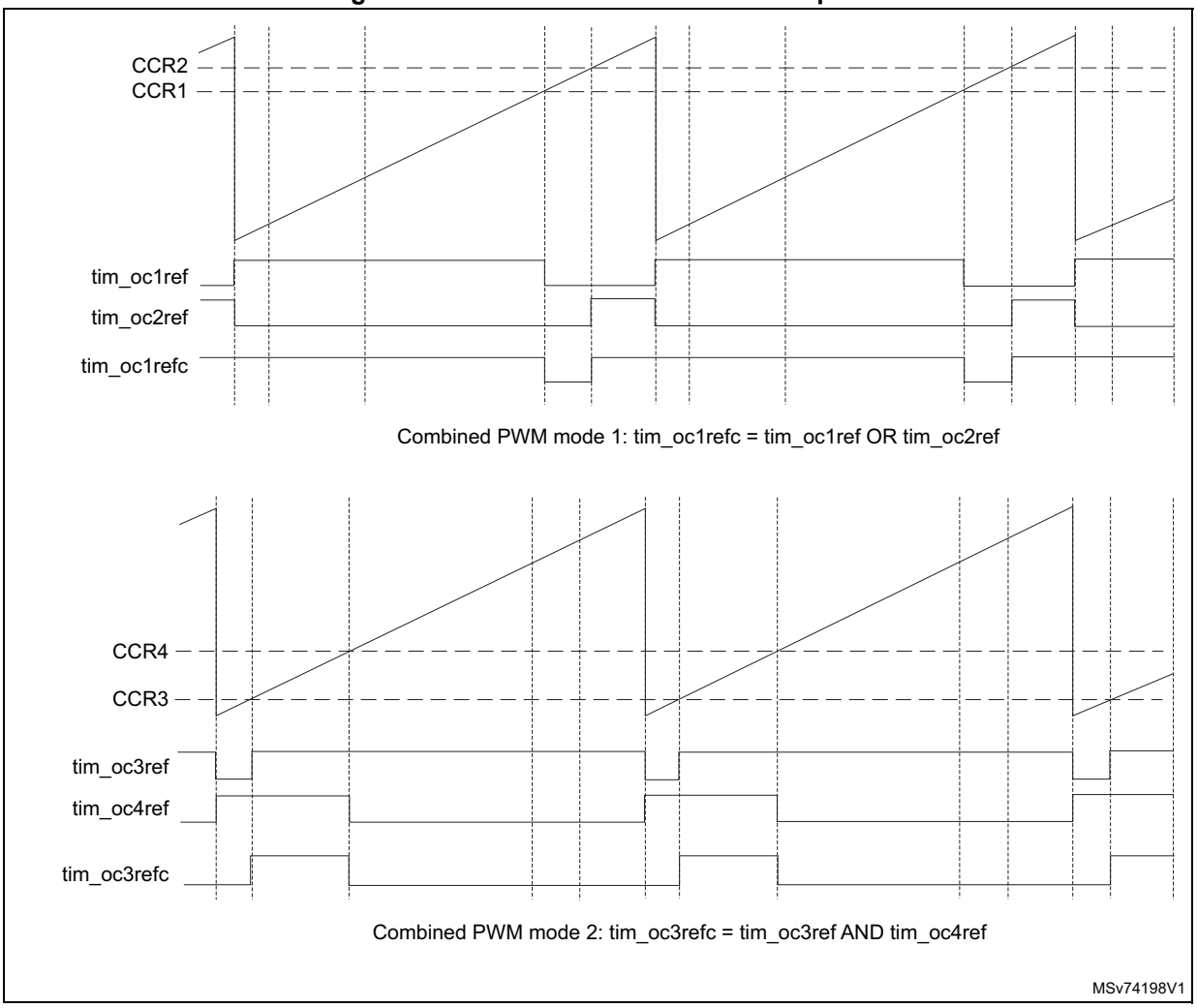
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in PWM mode 1 and the other in PWM mode 2).

Note: The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bits are not contiguous with the three least significant ones.

Figure 557 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.
- Channel 3 is configured in Combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.

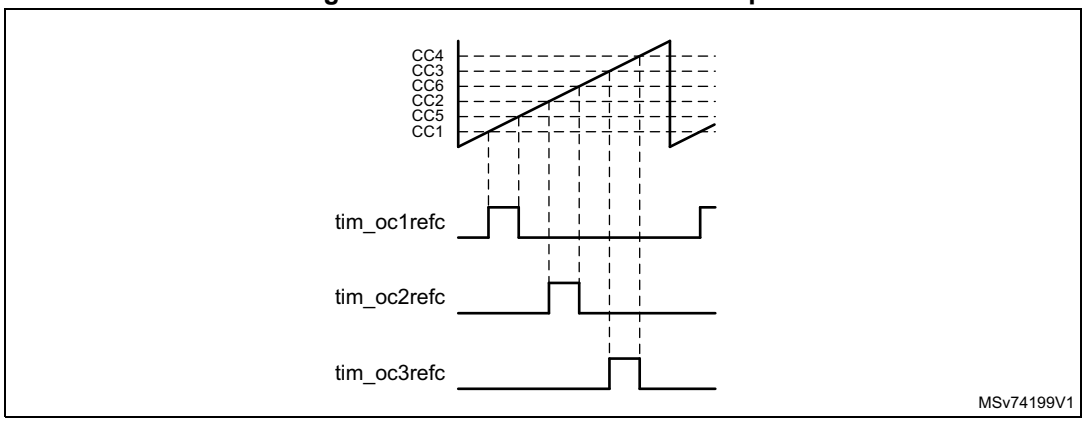
Figure 557. Combined PWM mode examples



The *Figure 558* represents an example suitable for a 3-phase interleaved converter requiring independent control of duty cycle and phase-shift. This is done using Combined PWM mode 2 and 4:

- Channel 1 is configured in Combined PWM mode 4
- Channel 2 is configured in Combined PWM mode 4
- Channel 3 is configured in Combined PWM mode 2

Figure 558. Combined PWM 3 example



34.3.16 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The tim_oc5ref signal is used to define the resulting combined signal.

The 4 bits GC5C[4:1] in the TIM_CCR5 allow selection on which reference signal the tim_oc5ref is combined. The resulting signals, tim_ocxrefc, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, tim_oc1refc is the logical AND of tim_oc1ref and tim_oc5ref
- If GC5C2 is set, tim_oc2refc is the logical AND of tim_oc2ref and tim_oc5ref
- If GC5C3 is set, tim_oc3refc is the logical AND of tim_oc3ref and tim_oc5ref
- If GC5C4 is set, tim_oc4refc is the logical AND of tim_oc4ref and tim_oc5ref

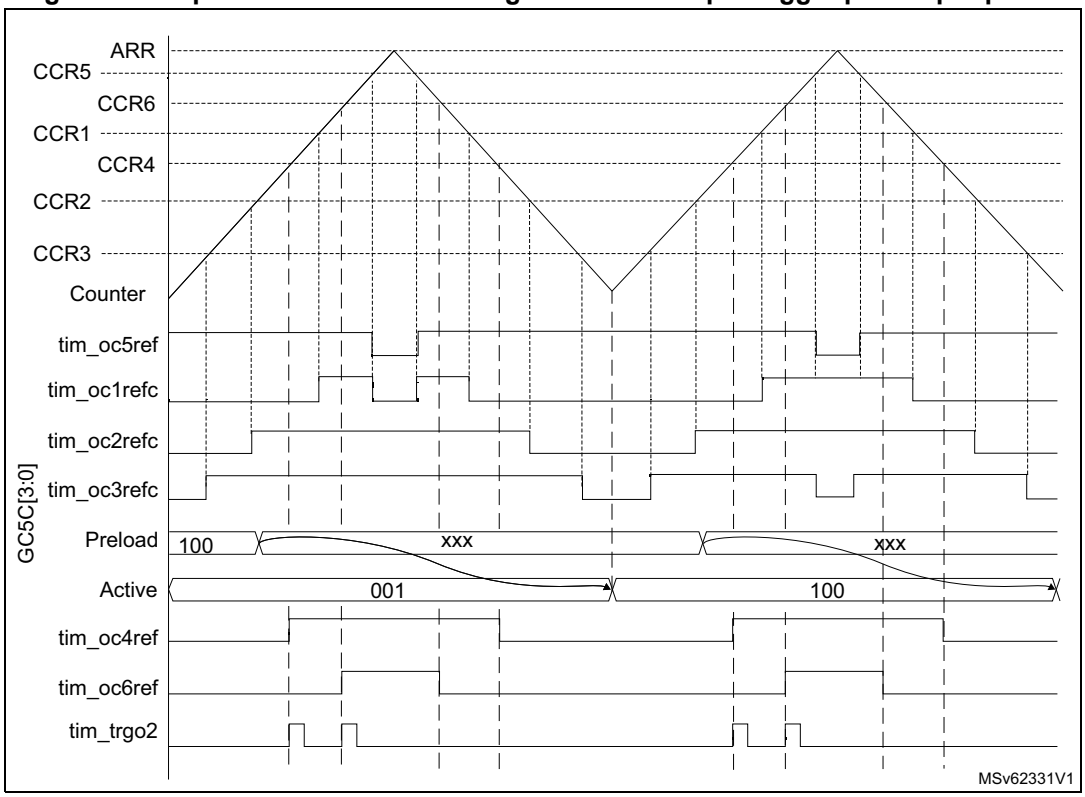
It is possible also to select an OR fonction with GC5C1O, GC5C2O, GC5C3O and GC5C4O bits, as following:

- If GC5C1O is set, tim_oc1refc is the logical OR of tim_oc1ref and tim_oc5ref
- If GC5C2O is set, tim_oc2refc is the logical OR of tim_oc2ref and tim_oc5ref
- If GC5C3O is set, tim_oc3refc is the logical OR of tim_oc3ref and tim_oc5ref
- If GC5C4O is set, tim_oc4refc is the logical OR of tim_oc4ref and tim_oc5ref

If both GC5Cx and GC5CxO bits are set, GC5Cx bit overrides GC5CxO programming and tim_ocxrefc is equal to tim_ocxref AND tim_oc5ref.

Combined 3-phase PWM mode can be selected independently on channels 1 to 4 by setting at least one of the bits GC5C[4:1] or GC5CO[4:1].

Figure 559. 3-phase combined PWM signals with multiple trigger pulses per period



The tim_trgo2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to *Section 34.3.36: ADC triggers* for more details.

34.3.17 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1/TIM8/TIM20) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters, or delays due to power switches).

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIM_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIM_CCER register and the MOE, OISx, OISxN, OSSI, and OSSR bits in the TIM_BDTR and TIM_CR2 registers. Refer to *Table 390: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a

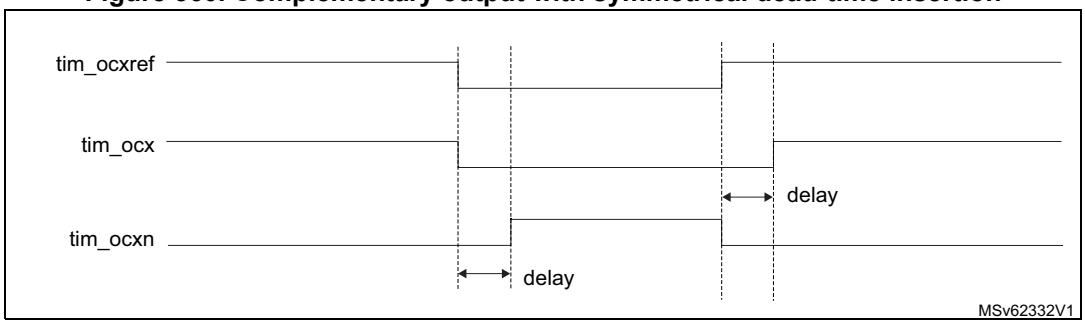
reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref considering CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1 in these examples.

Figure 560. Complementary output with symmetrical dead-time insertion



The DTAE bit in the TIM_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 561*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIM_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIM_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set, in the TIM_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 561. Asymmetrical deadtime

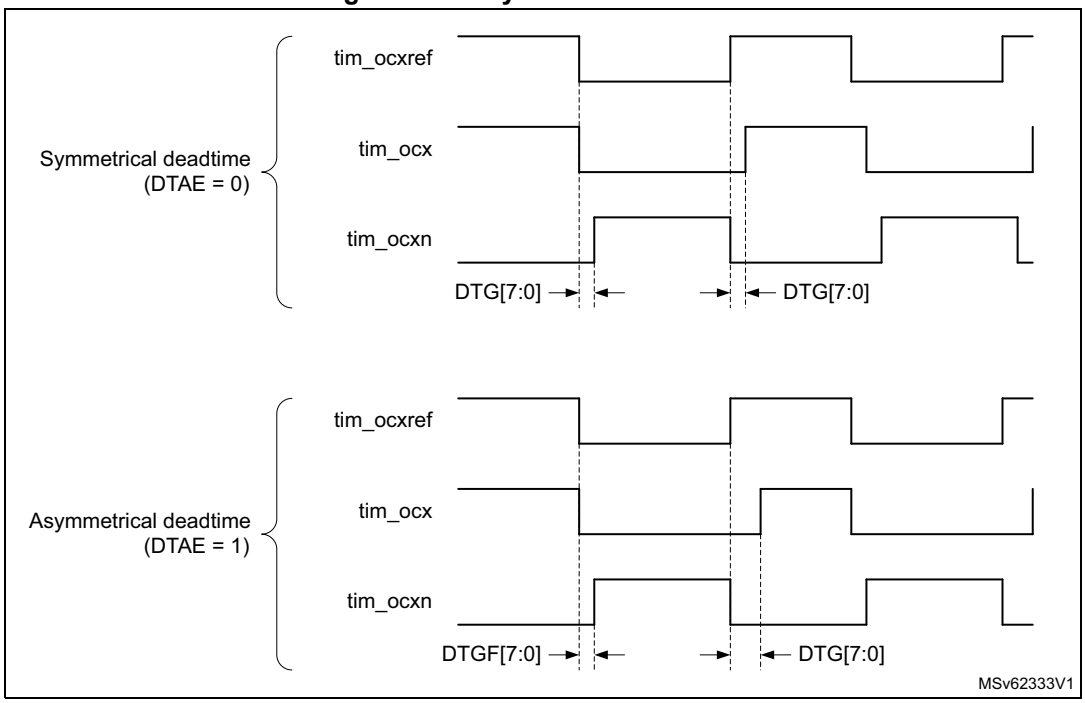


Figure 562. Dead-time waveforms with delay greater than the negative pulse

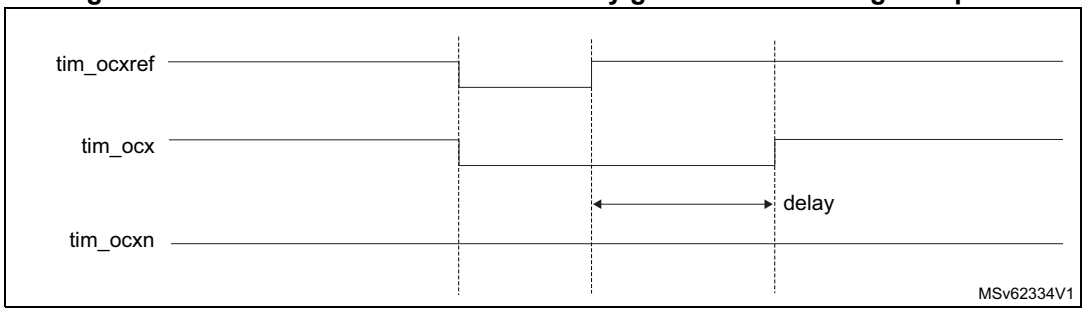
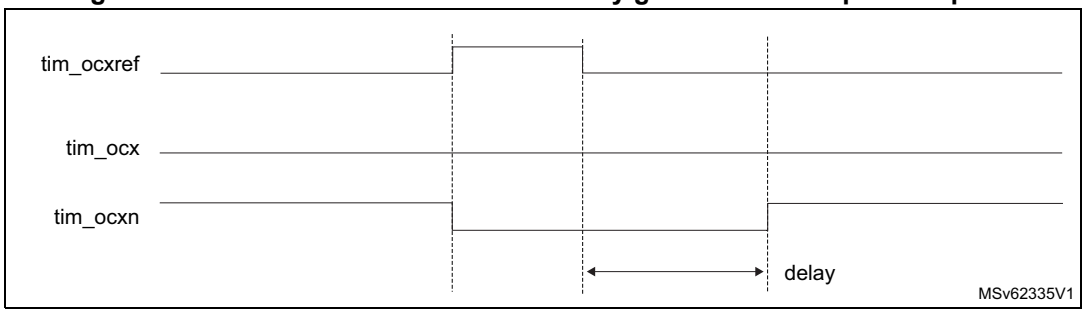


Figure 563. Dead-time waveforms with delay greater than the positive pulse



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIM_BDTR register. Refer to *Section 34.6.20: TIM break and dead-time register (TIM_BDTR)* for delay calculation.

Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIM_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

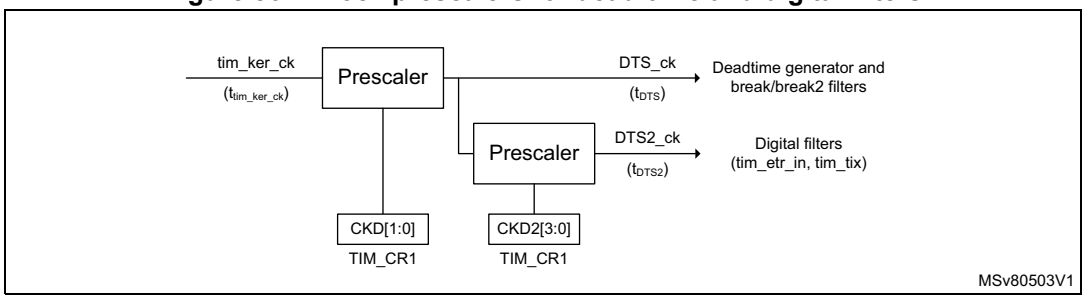
Note: When only tim_ocxn is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP = 0 then tim_ocxn = tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE = CCxNE = 1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

34.3.18 Clock dividers for dead-time generators and digital filters

The dead-time generators and the digital filters (for tim_etr_in, tim_tix) are using a DTS_ck clock (with a t_{DTS} period), derived from the tim_ker_ck clock.

A prescaler can decrease the frequency to extend the deadtime period programmed with the CKD[1:0] bitfield in the TIM_CR1 register. A second prescaler can decrease the resulting DTS_ck clock to extend solely the digital filters range, with the CKD2[3:0] bitfield in the TIM_CR1 register.

Figure 564. Clock prescalers for dead-time and digital filters



34.3.19 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, ECC/parity errors,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- The MOE bit in TIM_BDTR register is used to enable/disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the TIM_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- The OISx and OISxN bits in the TIM_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 390: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIM_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKEx and BKPx can be modified at the same time. When the BKEx and BKPx bits are written, a delay of one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIM_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIM_BKIN pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources:
  - coming from a tim_brk_cmpx input (refer to *Section 34.3.2: TIM1/TIM8/TIM20 pins and internal signals* for product specific implementation)
  - coming from a system break request (refer to *Section 34.3.2: TIM1/TIM8/TIM20 pins and internal signals* for product specific implementation)

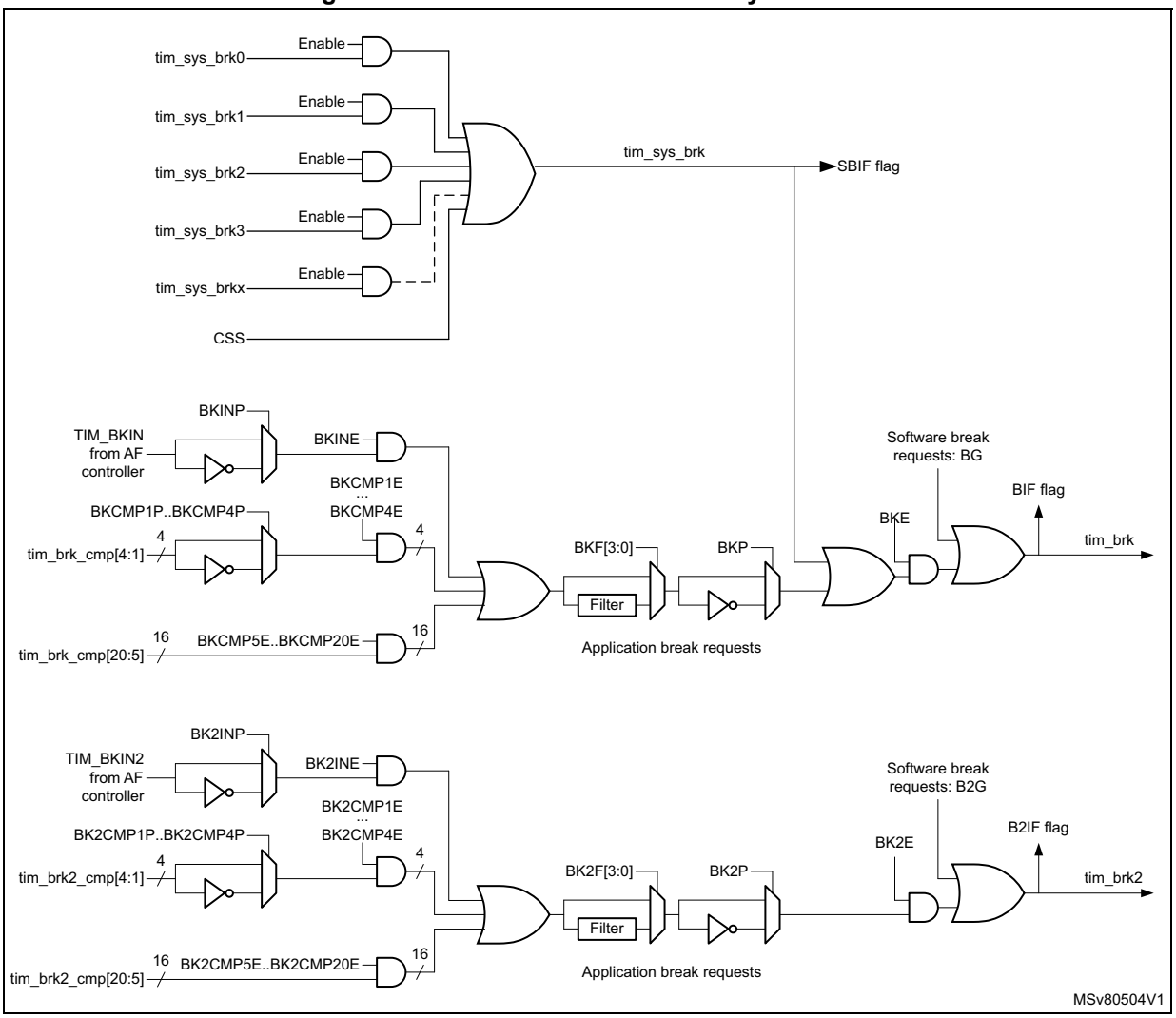
The sources for break2 (tim_brk2) are:

- External sources connected to one of the TIM_BKIN2 pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources coming from a tim_brk2_cmpx input (refer to *Section 34.3.2: TIM1/TIM8/TIM20 pins and internal signals* for product specific implementation)

Break events can also be generated by software using BG and B2G bits in the TIM_EGR register.

All sources are ORed before entering the timer tim_brk or tim_brk2 inputs, as per *Figure 565* below.

Figure 565. Break and Break2 circuitry overview



Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIM_CR2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to

their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 tim_ker_ck clock cycles).

  - If OSSI = 0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flags (B2GF, BGF, SBIF, BIF, and B2IF bits in the TIM_SR register) are set. An interrupt is generated if the BIE bit in the TIM_DIER register is set.
- An output disable status flag (ODS in the TIM_SR register) indicates the status of the outputs:
  - ODS = 0: Idle state. Break was triggered (or MOE was written to 0) and OIS states are forced on tim_ocx and tim_ocxn outputs
  - ODS = 1: Disabled state. Break2 was triggered, tim_ocx and tim_ocxn outputs are inactive
- If the AOE bit in the TIM_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors, or any security components.

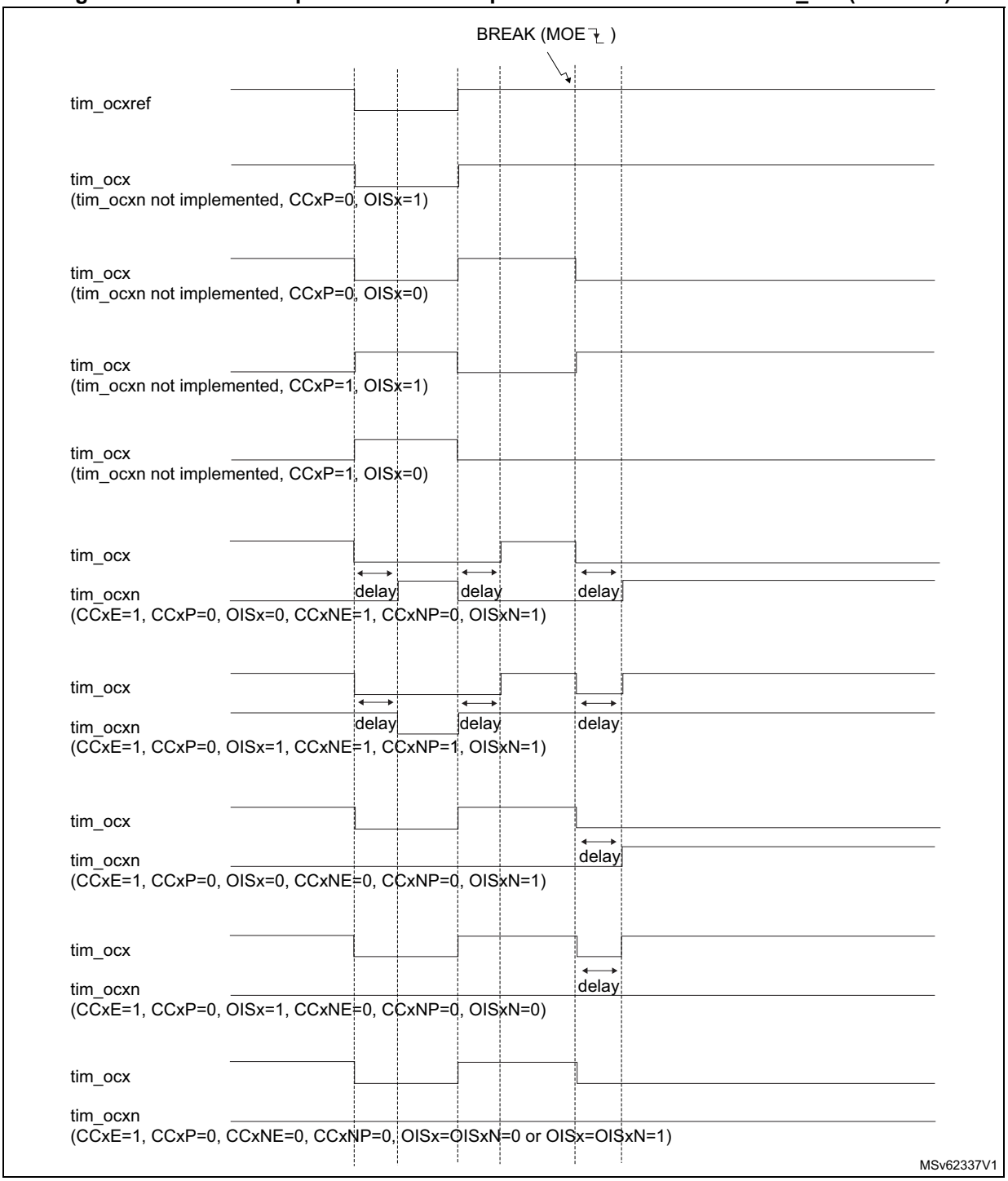
Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIM_CR2 register.

The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable, and polarity). The application can choose from three levels of protection selected by the LOCK bits in the TIM_BDTR register. Refer to *Section 34.6.20: TIM break and dead-time register (TIM_BDTR)*. The LOCK bits can be written only once after an MCU reset.

Figure 566 shows an example of behavior of the outputs in response to a break.

Figure 566. Various output behavior in response to a break event on tim_brk (OSSI = 1)



The two break inputs have different behaviors on timer outputs:

- The tim_brk input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- tim_brk2 can only disable (inactive state) the PWM outputs.

The tim_brk has a higher priority than tim_brk2 input, as described in *Table 383*.

The output disable status flag (ODS in the TIM_SR register) indicates the status of the outputs after a break event and is useful to distinguish break and break2 safe states.

Note: tim_brk2 must only be used with OSSR = OSSI = 1.

Table 383. Behavior of timer outputs versus tim_brk/tim_brk2 inputs

tim_brk	tim_brk2	Timer outputs state	Typical use case	
			tim_ocxn output (low side switches)	tim_ocx output (high side switches)
Active	X	– Inactive then forced output state (after a deadtime) – Outputs disabled if OSSI = 0 (control taken over by GPIO logic)	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

Figure 567 gives an example of tim_ocx and tim_ocxn output behavior in case of active signals on tim_brk and tim_brk2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIM_CCER register).

Figure 567. PWM output state following tim_brk and tim_brk2 assertion (OSSI = 1)

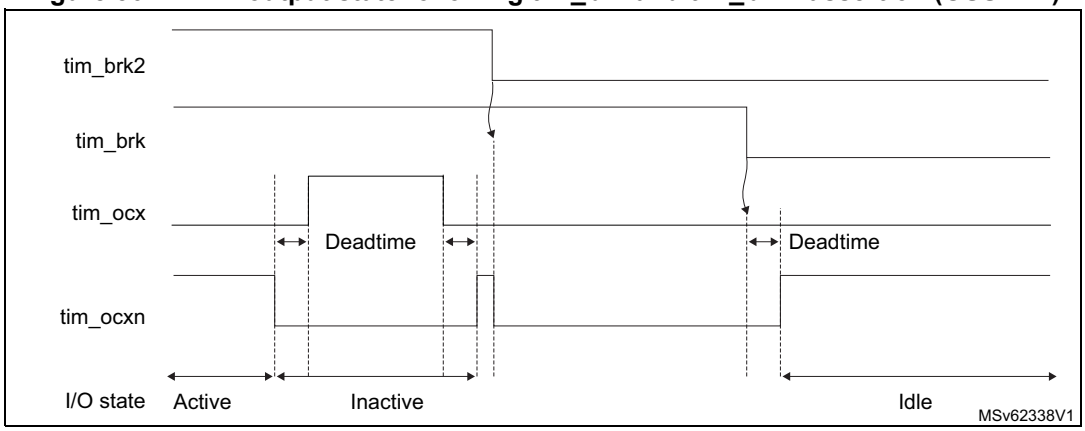
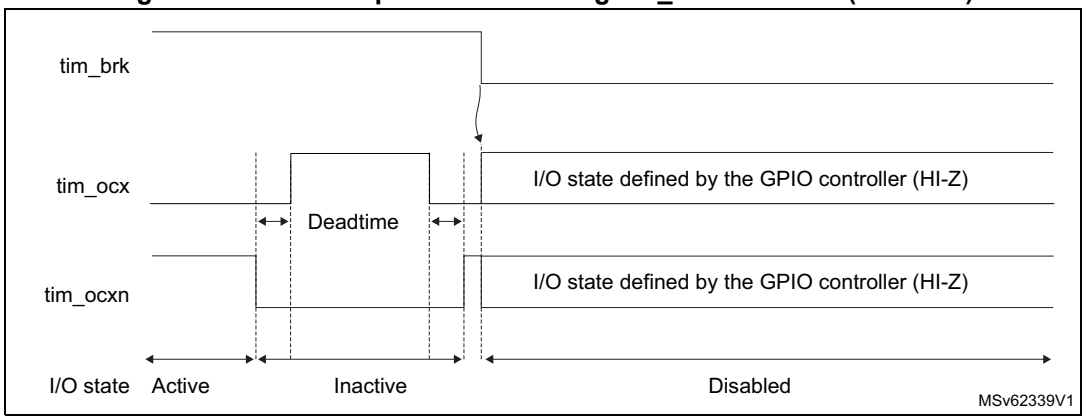


Figure 568. PWM output state following tim_brk assertion (OSSI = 0)



34.3.20 Delayed break

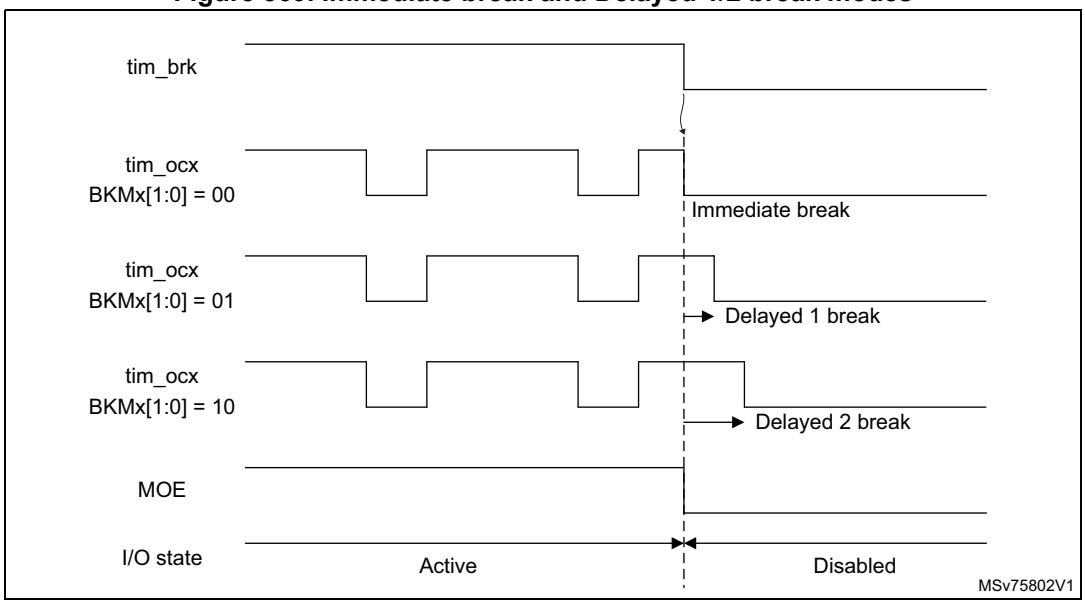
In Neutral-Point Clamped (NPC) multilevel converter topology, a break strategy with immediate outputs shut-down is not suitable, as it may damage the power switches with excessive voltage levels.

The delayed break features provide 2 options delaying the response to a break event, and implement a safe shut-down sequence.

BKMx[1:0] and BKMxN[1:0] bitfields in TIM_MPR1 provides 3 options for each complementary outputs (tim_oc1/tim_oc1n to tim_oc4/tim_oc4n), as show on *Figure 569* below:

- BKM[1:0] = 00: the break response is immediate
- BKM[1:0] = 01: Delayed 1 break. The break response is delayed with a delay defined in the DBK1[7:0] bitfield of the TIM_MPR2 register
- BKM[1:0] = 10: Delayed 2 break. The break response is delayed with a delay defined in the DBK2[7:0] bitfield of the TIM_MPR2 register

Figure 569. Immediate break and Delayed 1/2 break modes



The delayed break mode is triggered when:

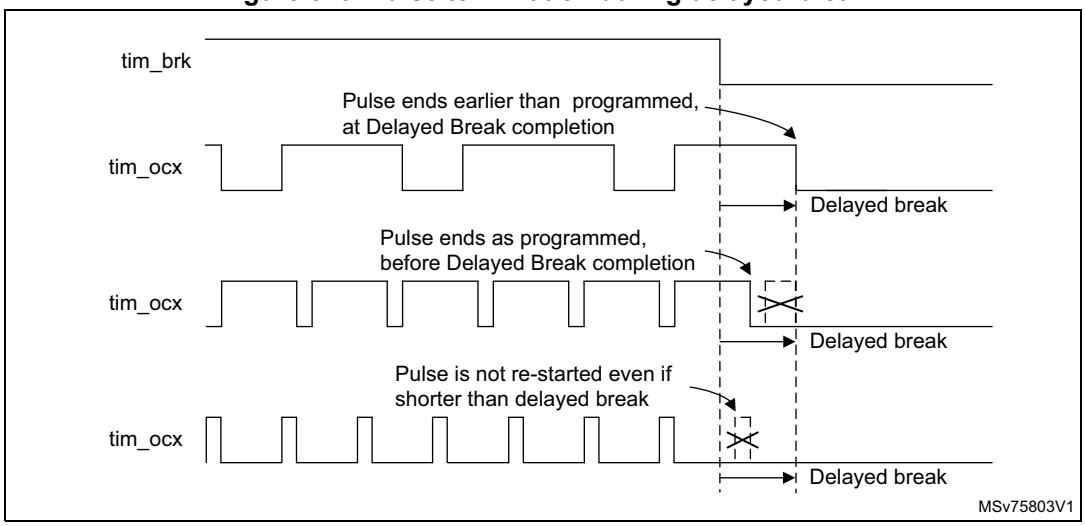
- A break event occurs
- The MOE bit is reset
- When the microcontroller enters debug mode with DBG_TIM_STOP=1

Warning: The break2 feature must be disabled when the delayed mode is enabled.

When delayed break mode is triggered, the MOE bit is immediately cleared, and cannot be set during the delayed break duration. In bidirectional break mode, the break output is reset immediately to indicate the delayed break start to external components. Once the longest of the 2 delays is over, the deadtime generator is enabled to prevent any deadtime violation if OISx/OISxN bits are set.

If the tim_ocx/tim_ocxn signal is reset before the end of the delayed break (due to a compare match in PWM mode, typically), the signal is not maintained during the full delayed break period, to maintain the duty cycle, as shown on *Figure 570* below. Still, the Delayed break circuitry guarantees that the tim_ocx/tim_ocxn signals cannot be set again once the break event occurred.

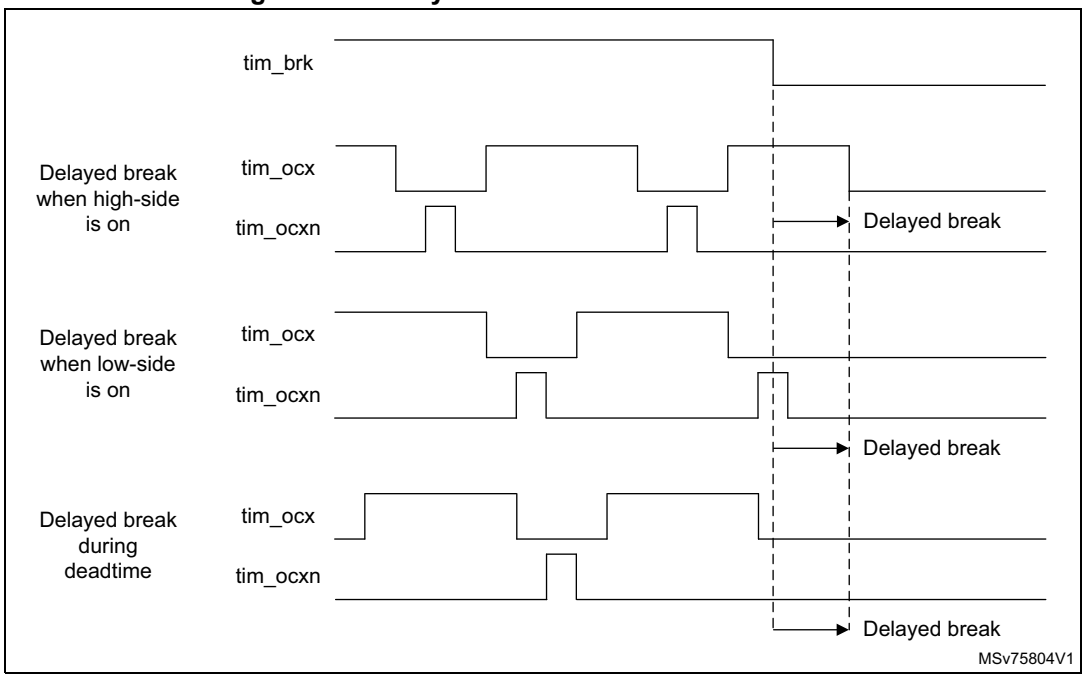
Figure 570. Pulse termination during delayed break



The Figure 563 depicts the behavior when the deadtime generator is enabled, for the 3 cases:

– Delayed break triggered when tim_ocx is set
– Delayed break triggered when tim_ocxn is set
– Delayed break triggered during deadtime interval

Figure 571. Delayed break with deadtime insertion



The delay programming is identical to deadtime generator's, with 2 user-programmable durations in the TIM_MPR2 register: DBK1[7:0] bitfield for the Delayed 1 break and DBK2[7:0] bitfield for the Delayed 2 break.

DBK1[7:0] and DBK2[7:0] bitfields are preloaded: they can be changed during timer operation, and the new value is transferred to the active register when an update event occurs.

DBK1[7:0] and DBK2[7:0] bitfields are partially preloaded: they can be changed during timer operation, and the new value is transferred to the active register when an update event occurs.

The bits coding for t_{DTG} selection are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBKx[6:0] if DBKx[7] = 0
- DBKx[5:0] if DBKx[7:6] = 10
- DBKx[4:0] if DBKx[7:5] = 110 or 111

The BKMx[1:0] and BKMxN[1:0] bits are static, they can only be modified when the counter is stopped (CEN=0 in TIM_CR1).

The content of the TIM_MPR1 and TIM_MPR2 register can be write protected using the LOCK[1:0] bits in the TIM_BDTR register.

34.3.21 Bidirectional break inputs

The TIM1/TIM8/TIM20 feature bidirectional break I/Os, as represented on *Figure 572*.

This provides support for:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk and tim_brk2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIM_BDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIM_BDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the tim_brk and tim_brk2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (for example CSS), from on-chip peripherals, or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to 0 to indicate to the external components that the timer is entered in break state. However, this is valid only if the break is enabled (BKE or B2KE = 1). When a software break event is generated with BKE or B2KE = 0), the outputs are put in safe state and the break flag is set, but there is no effect on the TIM_BKIN and TIM_BKIN2 I/Os.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM (BK2DSRM) bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 384*).

Table 384. Break protection disarming conditions

MOE	BKBID (BK2BID)	BKDSRM (BK2DSRM)	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and rearming break circuitry

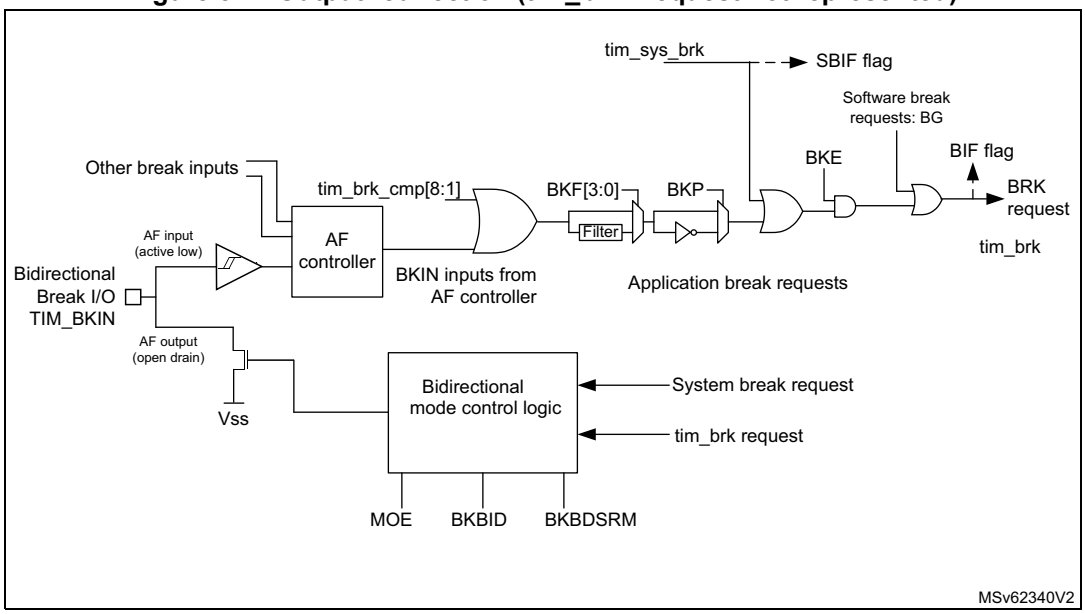
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before rearming).
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 572. Output redirection (tim_brk2 request not represented)



34.3.22 Output idle state override

It is possible to override the outputs states when they are disabled (ie MOE bit reset in TIM_BDTR). This feature can be used to apply specific patterns prior to start PWM operations, for instance to refresh bootstrap capacitors of power switch gate drivers.

The OOC control bit in the TIM_OOR register acts as following:

- OOC= 0: 0: tim_ocx and tim_ocxn outputs are in the default idle state
- OOC=1: tim_ocx and tim_ocxn outputs are forced in an idle state defined with OOSx/OOSxN bits in the TIM_OOR register

The OOC bit can only be modified when the outputs are in idle state (MOE = 0). It is cleared by hardware when the MOE bit is set, and the deadtime generator ensures there's no deadtime violation on complementary outputs.

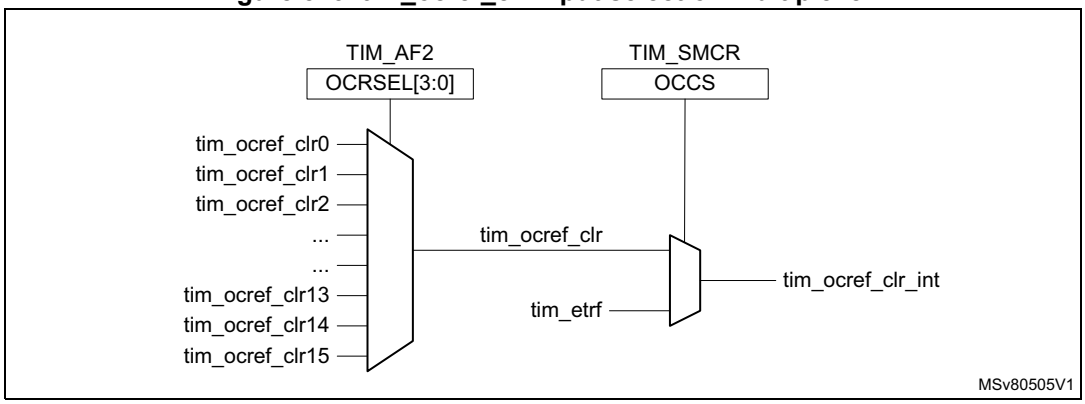
The OOSx/OOSxN bits in the TIM_OOR register cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

34.3.23 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIM_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It must not be used in Forced mode. tim_ocref_clr_int input can be selected between the tim_ocref_clr input and tim_etrf (tim_etr_in after the filter) by configuring the OCCS bit in the TIM_SMCR register.

The tim_ocref_clr input can be selected among several inputs, using the OCRSEL[3:0] bitfield in the TIM_AF2 register, as shown on the *Figure 573* below. Refer to *Section 34.3.2: TIM1/TIM8/TIM20 pins and internal signals* for a list of sources available in the product.

Figure 573. tim_ocref_clr input selection multiplexer

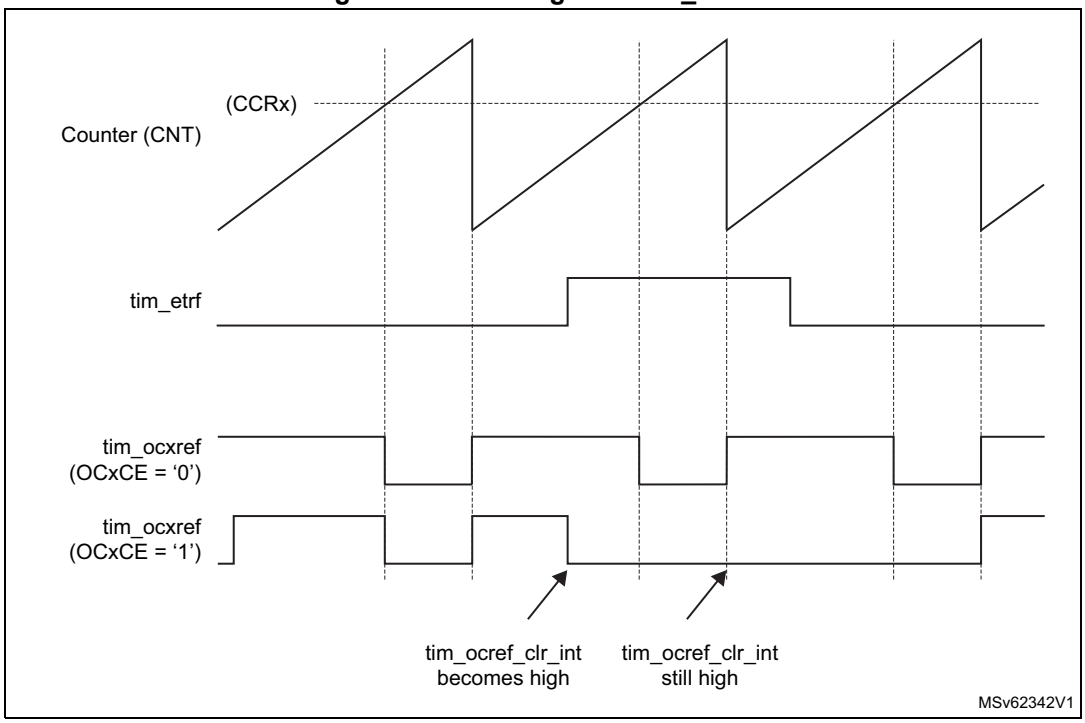


When tim_etrf is chosen, tim_etr_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] of the TIM_SMCR register set to 00.
2. The external clock mode 2 must be disabled: bit ECE of the TIM_SMCR register set to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to application needs (as per polarity of the source connected to the trigger and eventual need to remove noise using the filter).

Figure 574 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Figure 574. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), then tim_ocxref is enabled again at the next counter overflow.

This feature is also supported in combined and asymmetric modes. In this case, it is mandatory to have the OCxCE bit enabled for the channel, the OCxCE bit of the complementary channel (e.g. OC2CE = X for oc2ref in *Figure 575*) is ignored in this case.

Figure 575 to *Figure 578* are describing the output waveforms for some use cases, when the ocref_clr_int signal is asserted.

Figure 575. Asymmetric PWM mode 1

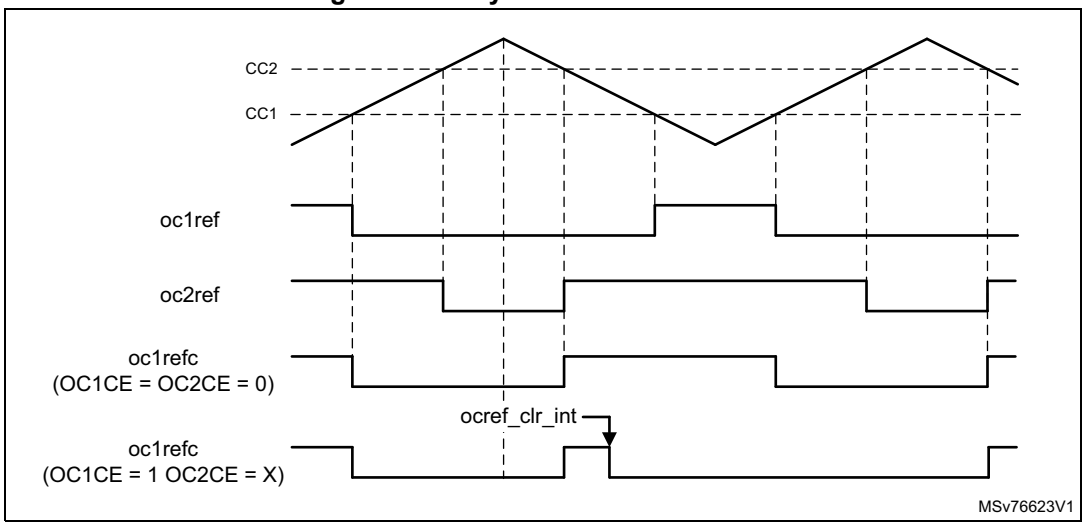


Figure 576. Asymmetric PWM mode 2

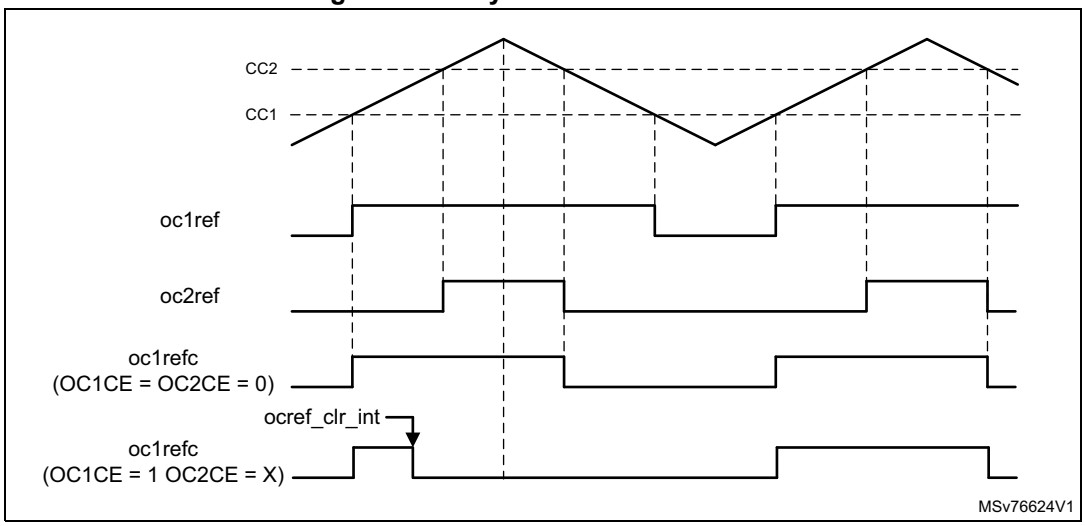


Figure 577. Asymmetric PWM mode 7 (OC1refc = oc1ref OR oc5ref)

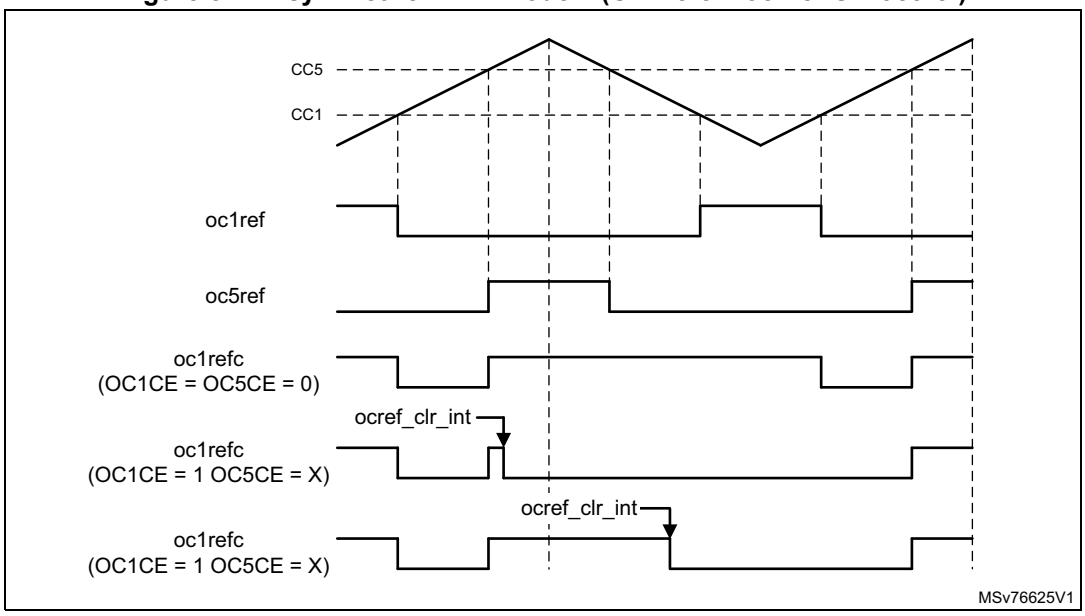
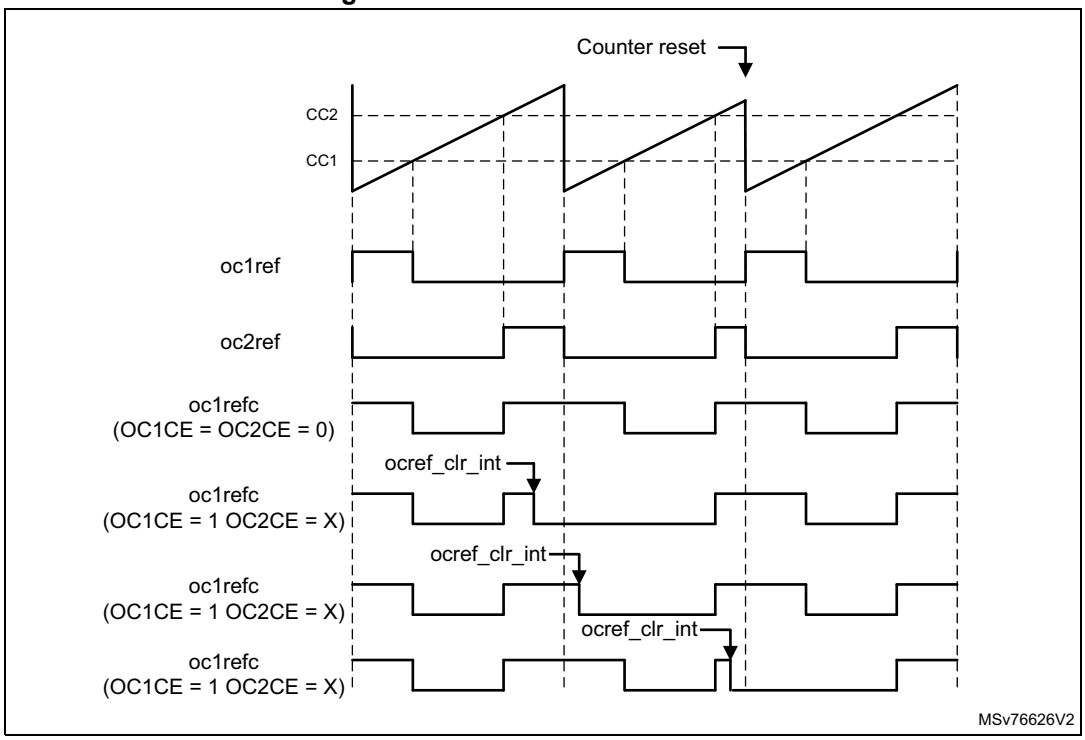


Figure 578. Combined PWM mode 1



Conditions of use

This feature is intended to support cycle-by-cycle current control loops, typically with a comparator shortening the PWM on-time when a current threshold is exceeded.

It cannot be used for the following modes, where the OCxCE bit must be cleared:

- PWM toggle mode (OCxM[4:0] = 0x03)
- One pulse mode with PWM active on match (OPM=1 and OCxM[4:0] = 0x06, 0x08, 0x0C, 0x0E, 0x10, 0x12, 0x14, 0x16, 0x18)
- Combined 3-phase modes (GC5Cx or GC5CxO = 1)
- Combined PWM mode 1 or 3 in center aligned mode (CMS[1:0] = 1,2,3 and OCxM[4:0] = 0x0C or 0x10)

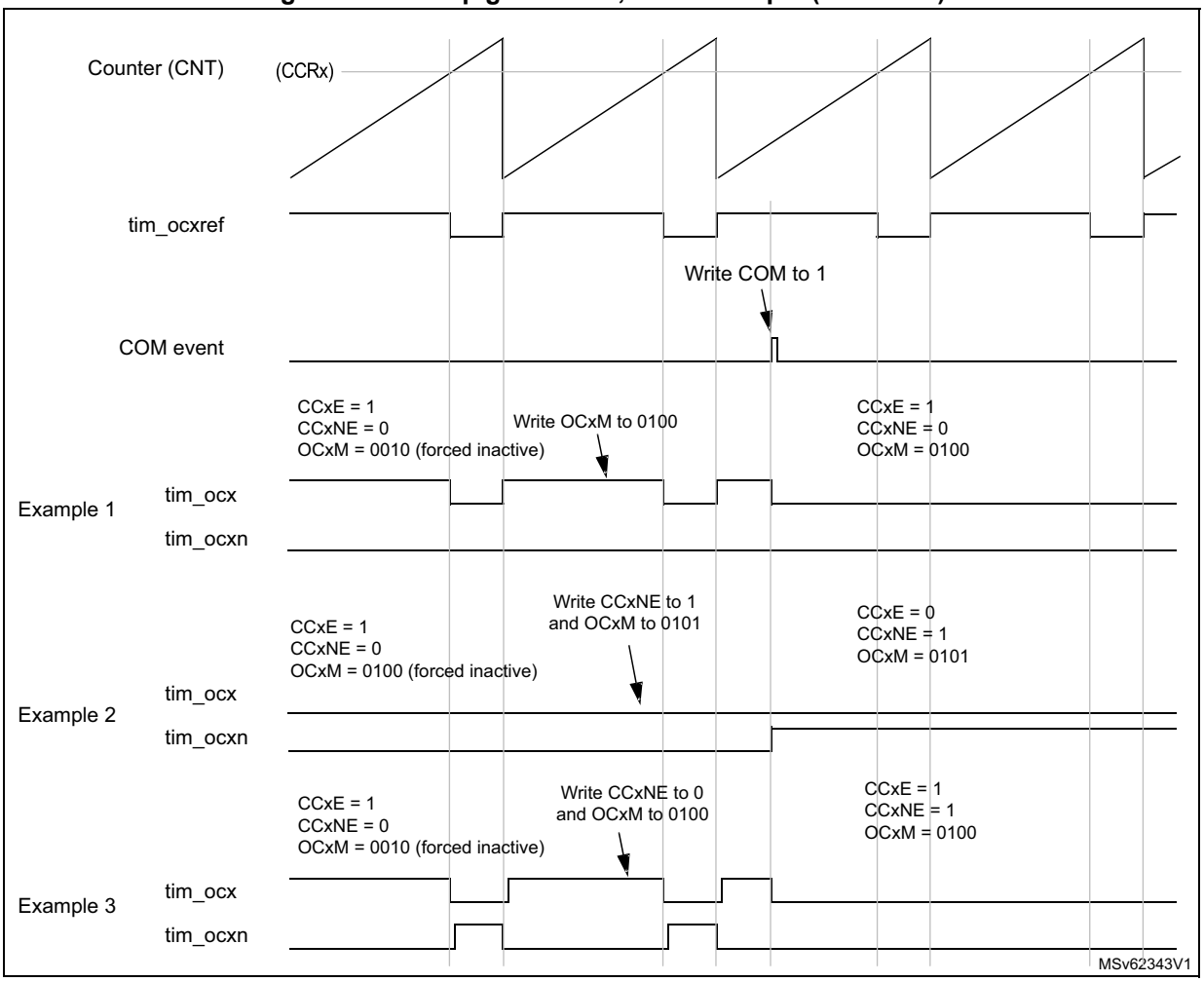
34.3.24 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE, and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIM_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIM_SR register), which can generate an interrupt (if the COMIE bit is set in the TIM_DIER register) or a DMA request (if the COMDE bit is set in the TIM_DIER register).

Figure 579 describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in three different examples of programmed configurations.

Figure 579. 6-step generation, COM example (OSSR = 1)



34.3.25 One-pulse mode

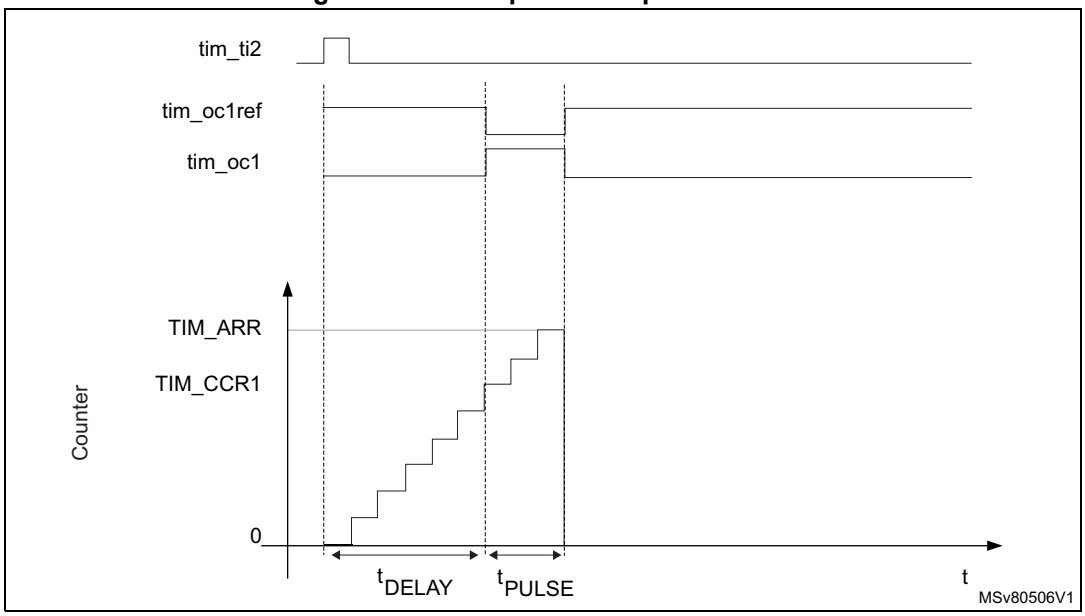
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIM_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 580. Example of one pulse mode.



In the following example, the user wants to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

- Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIM_CCMR1 register.
- tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
- Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIM_SMCR register.
- tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIM_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIM_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIM_ARR - TIM_CCR1).
- Suppose the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the auto-reload value. This is achieved by enabling PWM mode 2 (OC1M = 00111 in TIM_CCMR1). Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIM_CCMR1 register and ARPE in the TIM_CR1 register. In this case one has to write the compare value in the TIM_CCR1 register, the autoreload value in the TIM_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

In this example, the DIR and CMS bits in the TIM_CR1 register must be low.

Since only one pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIM_CR1 register to stop the counter at the next update event (when the counter rolls over

from the autoreload value back to 0). When OPM bit in the TIM_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be achieved.

To output a waveform with the minimum delay, the OCxFE bit can be set in the TIM_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

34.3.26 Retriggerable One-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with nonretriggerable one-pulse mode described in *Section 34.3.25*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

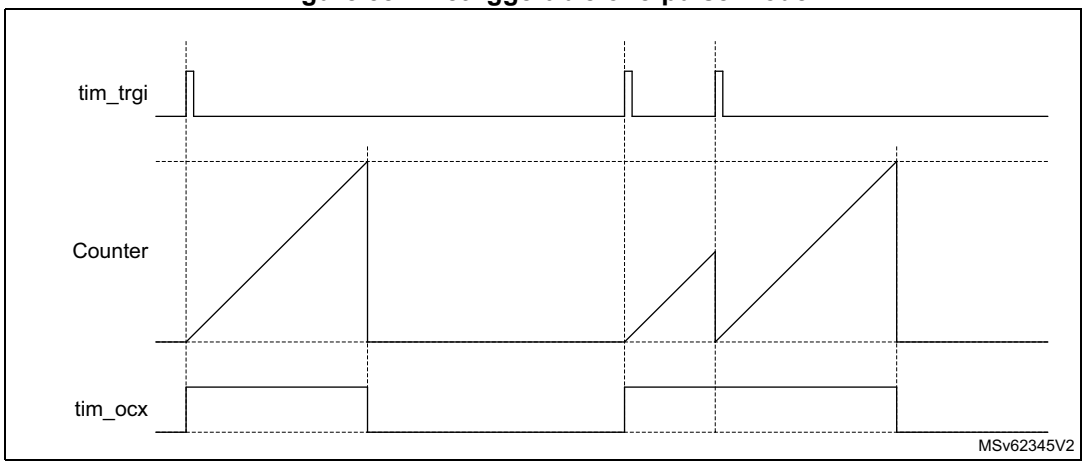
The timer must be in Slave mode, with the bits SMS[4:0] = 01000 (Combined Reset + trigger mode) in the TIM_SMCR register, and the OCxM[4:0] bits set to 01000 or 01001 for retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[4:0] and SMS[4:0] bitfields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIM_CR1.

Figure 581. Retriggerable one-pulse mode



34.3.27 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulsewidth generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on *Figure 582*.

Figure 582. Pulse generator circuitry

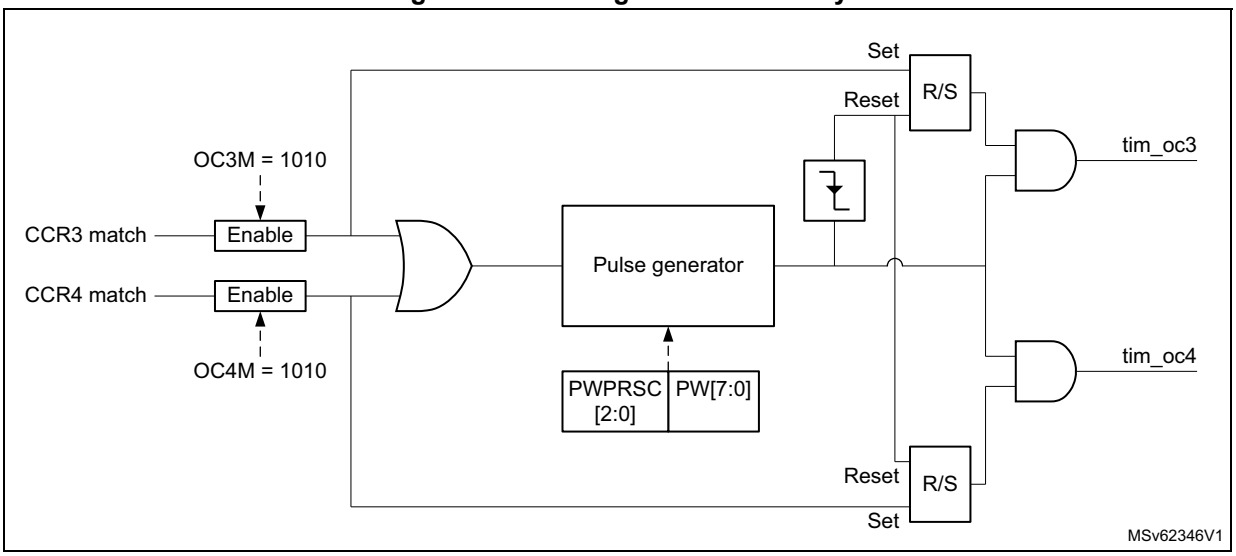
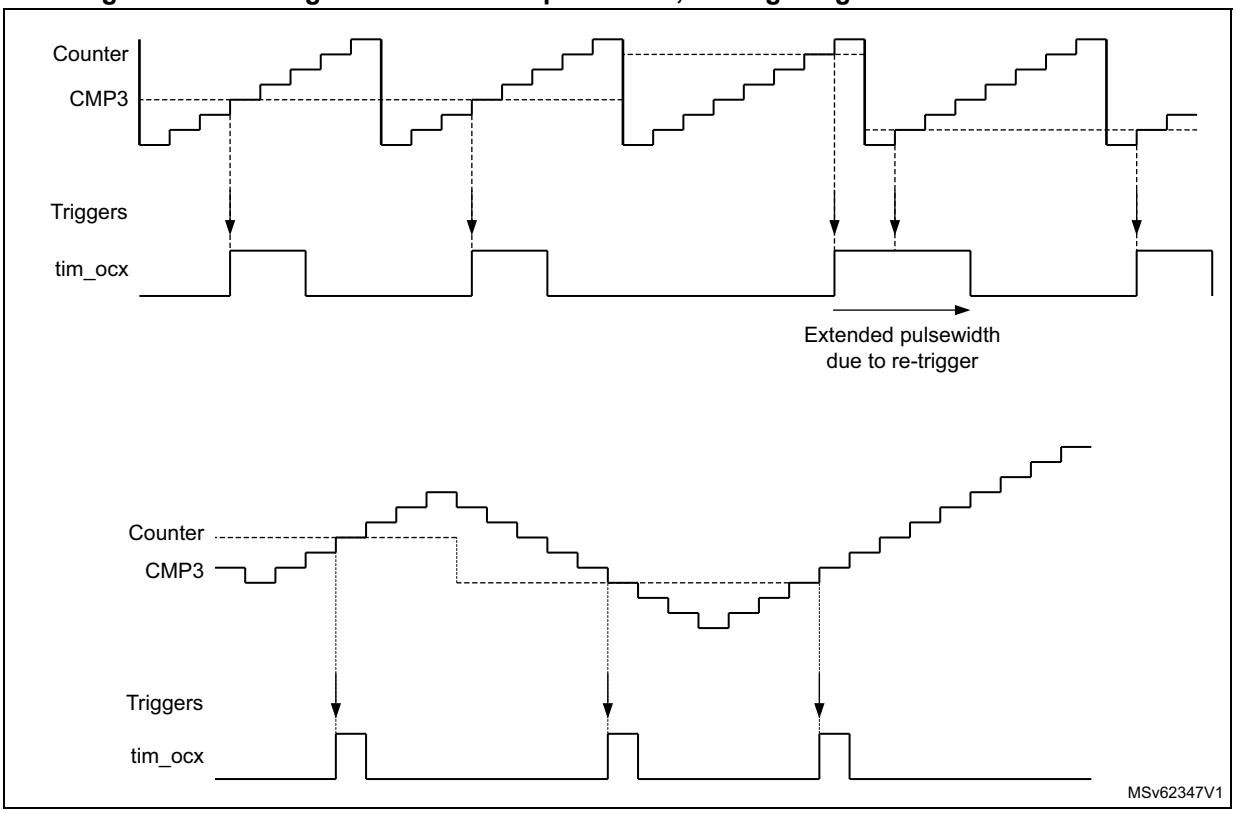


Figure 583 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 583. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bitfields in TIM_CCMR2 register.

The pulsewidth is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

t_{PW} = PW[7:0] x t_{PWG}

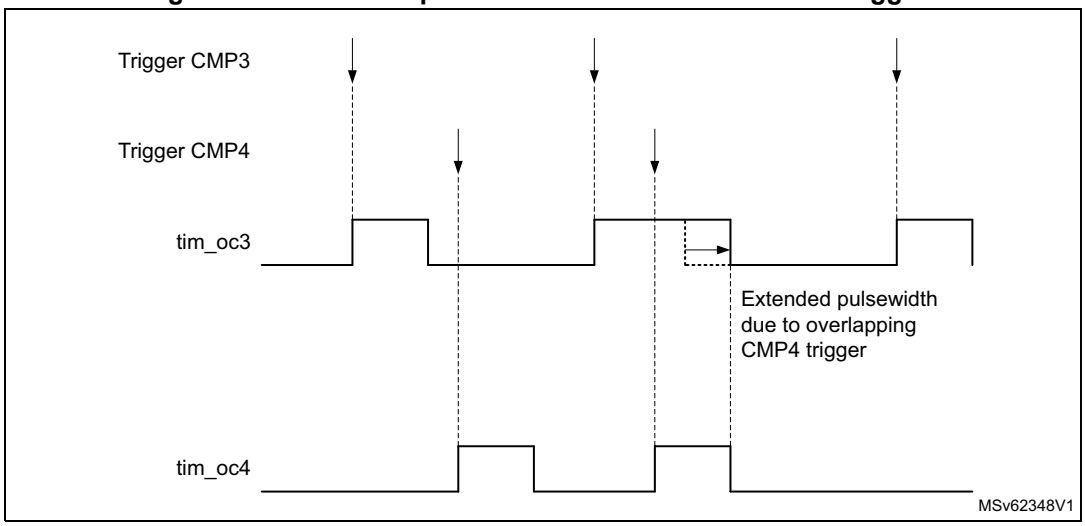
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 584).

Figure 584. Extended pulsewidth in case of concurrent triggers



34.3.28 Encoder interface mode

Quadrature encoder

To select Encoder Interface mode write SMS = 0001 in the TIM_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 0010 if it is counting on tim_ti2 edges only and SMS = 0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIM_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs tim_ti1 and tim_ti2 are used to interface to an quadrature encoder. Refer to *Table 385*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIM_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIM_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIM_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIM_ARR must be configured before starting. In the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 385. Counting direction versus encoder signals (CC1P = CC2P = 0)

Active edge	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	01110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	01111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	00001	High	Down	Up	No count	No count
		Low	Up	Down	No count	No count
Counting on tim_ti2 only x2 mode	00010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	00011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

Figure 585 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example the configuration is the following:

- CC1S = 01 (TIM_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIM_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P = 0 and CC1NP = 0 (TIM_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P = 0 and CC2NP = 0 (TIM_CCER register, tim_ti1fp2 noninverted, tim_ti1fp2= tim_ti2).
- SMS = 0011 (TIM_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIM_CR1 register, Counter enabled).

Figure 585. Example of counter operation in encoder interface mode.

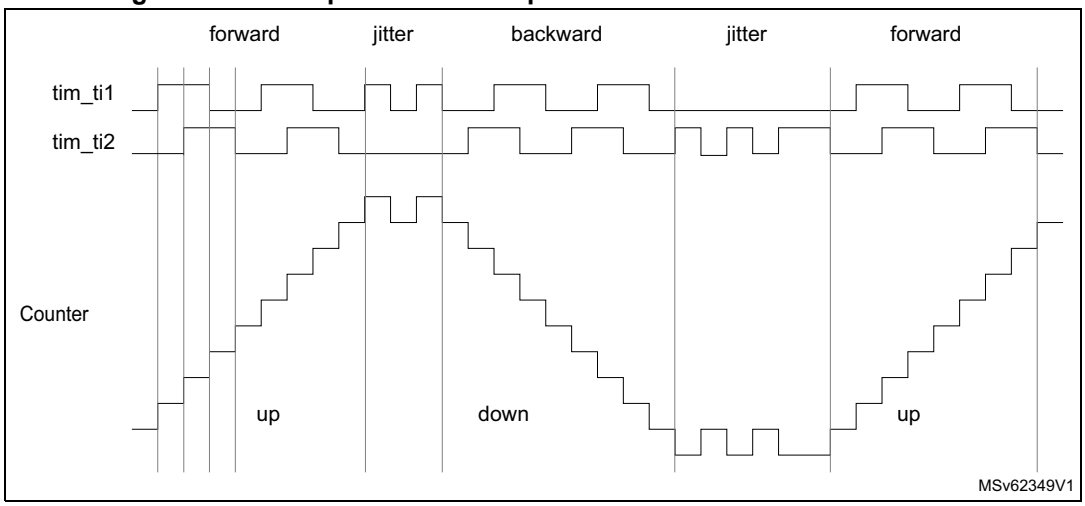


Figure 586 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 586. Example of encoder interface mode with tim_ti1fp1 polarity inverted.

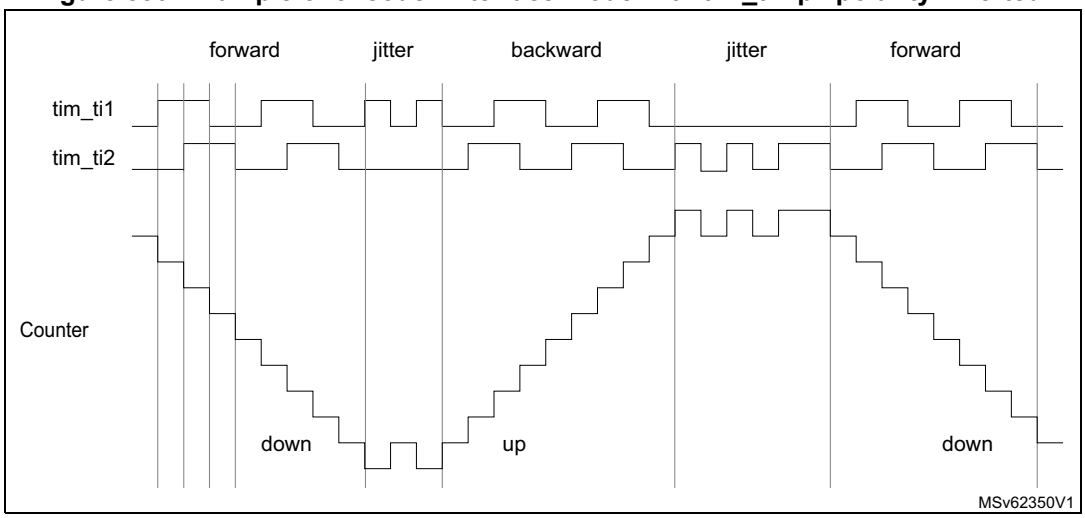
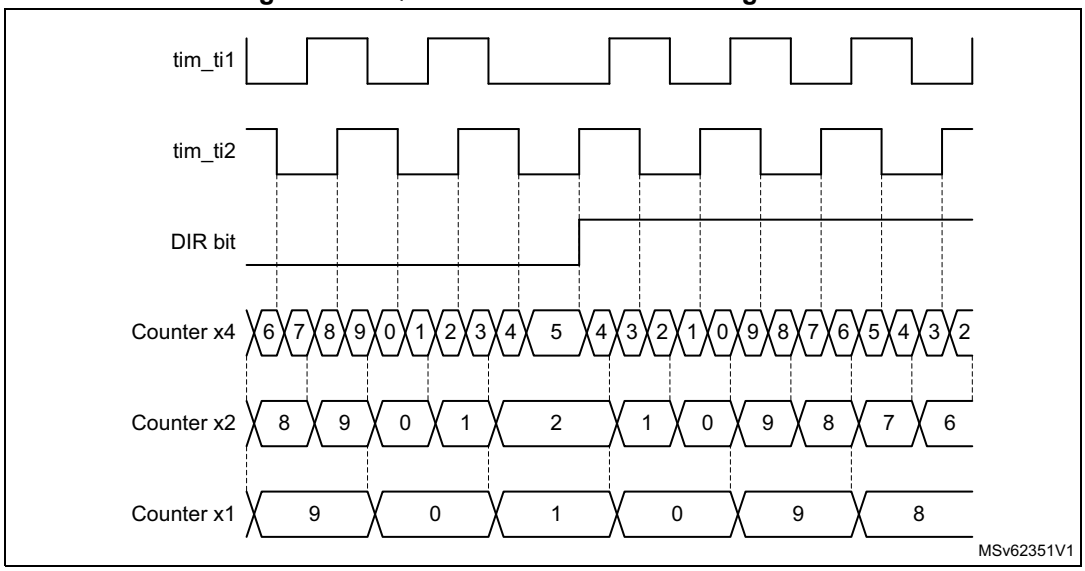


Figure 587 shows the timer counter value during a speed reversal, for various counting modes.

Figure 587. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIM_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

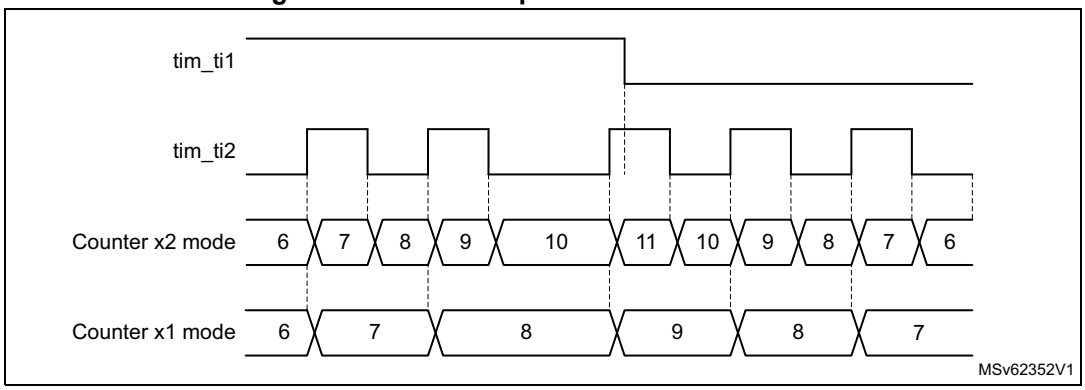
In the clock plus direction mode shown on *Figure 588*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01010: x2 mode, the counter is updated on both rising and falling edges of the clock
- 01011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 588. Direction plus clock encoder mode



Directional clock encoder mode

In the directional clock mode on *Figure 589*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 589*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 590*).
- 01101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 589*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 590*).

Figure 589. Directional clock encoder mode (CC1P = CC2P = 0)

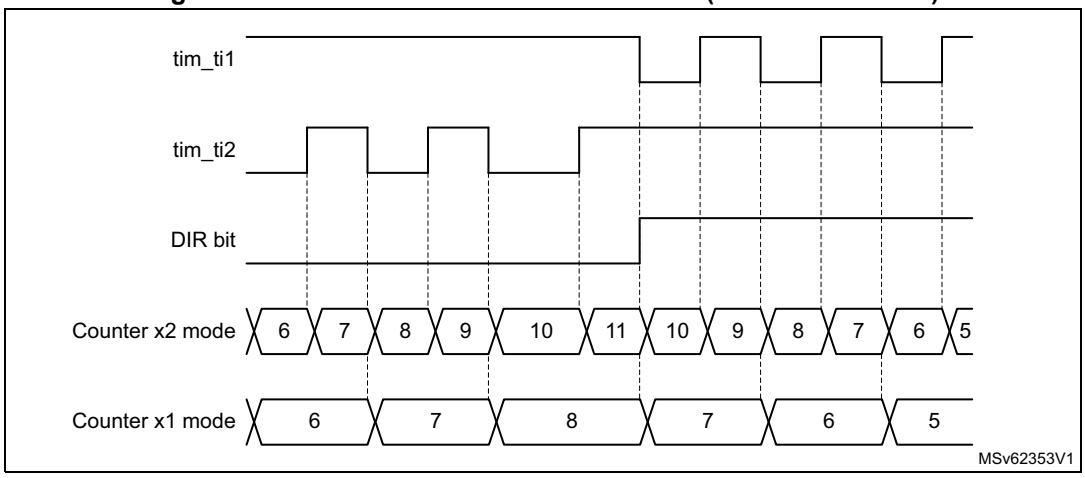


Figure 590. Directional clock encoder mode (CC1P = CC2P = 1)

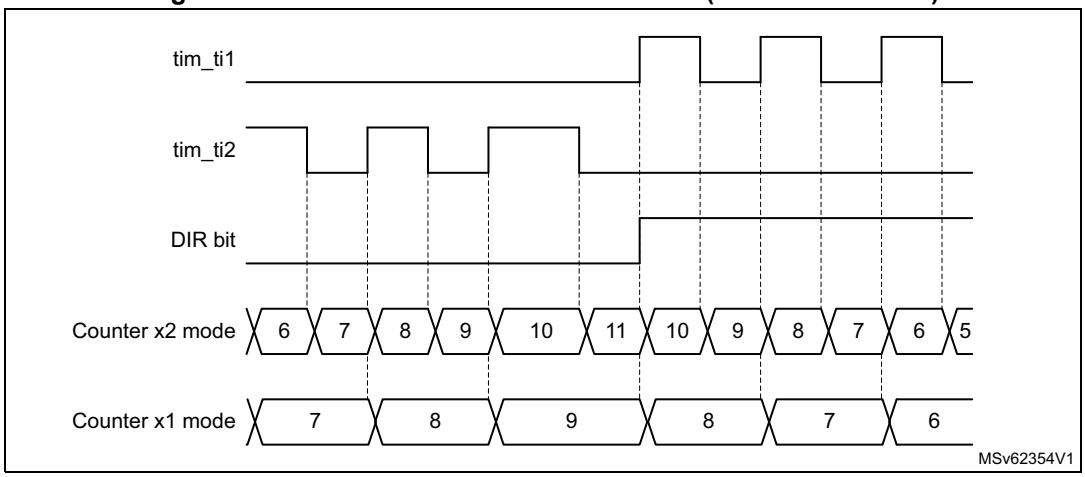


Table 386 here-below details how the directional clock mode operates, for any input transition.

Table 386. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	01100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	01100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	01101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	01101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index input

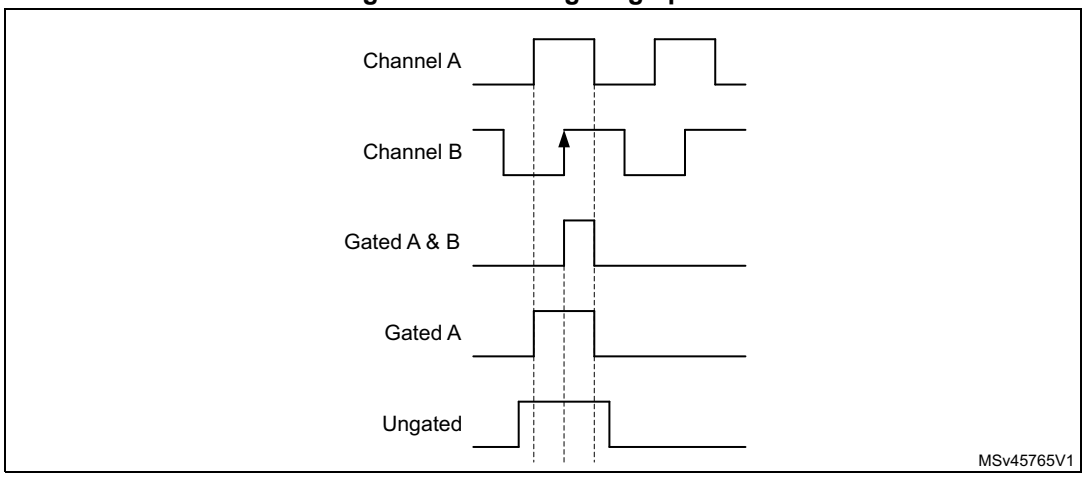
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIM_ECR register. The IE bit must be set only in encoder mode, when the SMS[4:0] bitfield has the following values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111.

Available encoders are proposed with several options for index pulse conditioning, as per *Figure 591*:

- gated with A and B: the pulsewidth is 1/4 of one channel period, aligned with both A and B edges
- gated with A (or gated with B): the pulsewidth is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B)
- ungated: the pulsewidth is up to one channel period, without any alignment to the edges

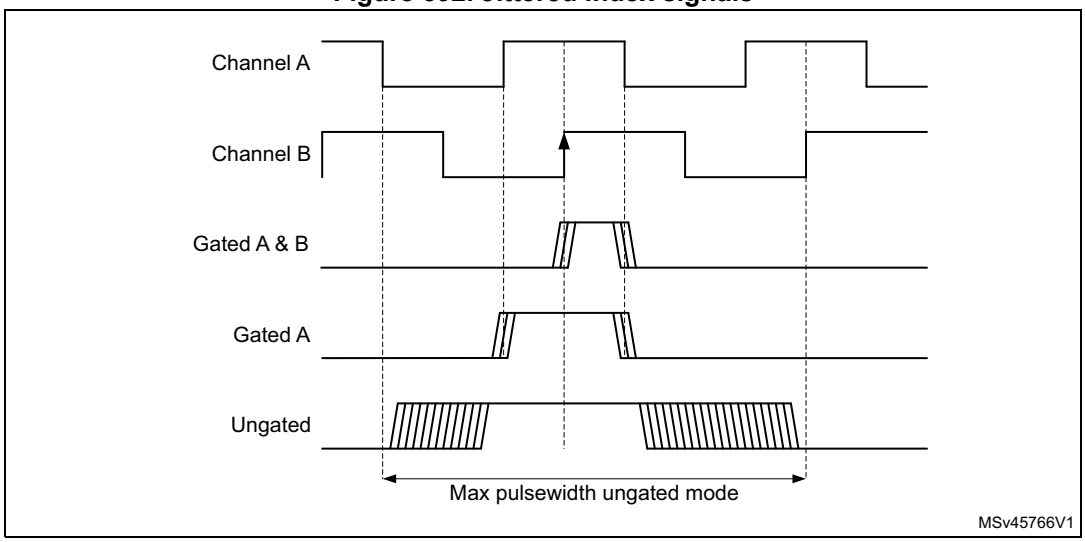
Figure 591. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as show on *Figure 592*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulsewidth is greater or equal to two encoder period, the counter is reset multiple times.

Figure 592. Jittered Index signals



The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example channel A and

channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIM_ECR register.

The index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIM_ARR when down counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. *Figure 593* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 593. Index generation for IPOS[1:0] = 11

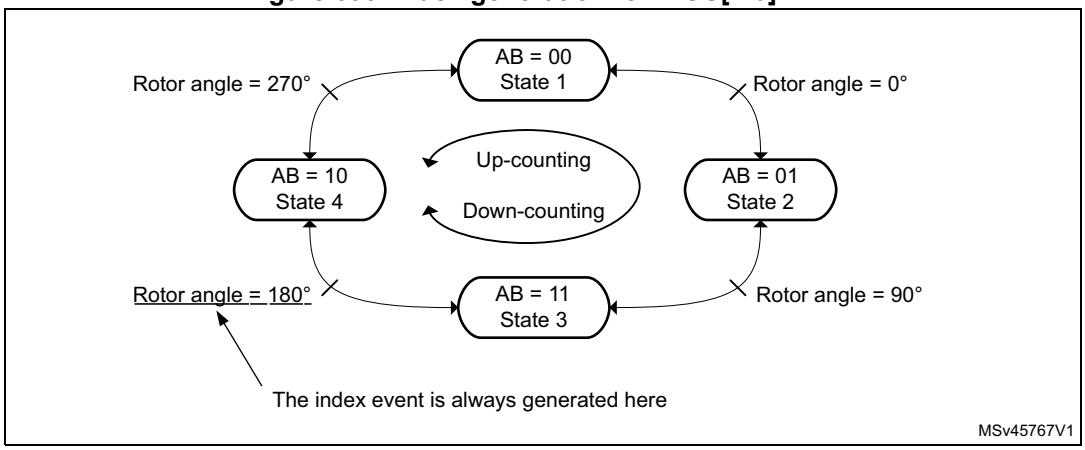


Figure 594 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is 11 (ChA = 1, ChB = 1), when up-counting (DIR bit = 0).
- Counter set to TIM_ARR when exiting the 11 state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

Figure 594. Counter reading with index gated on channel A (IPOS[1:0] = 11)

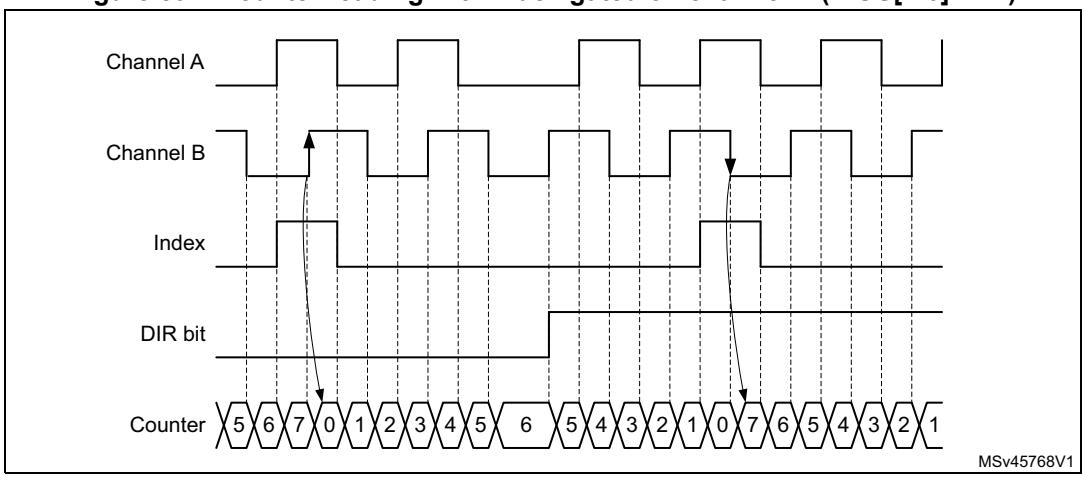


Figure 595. presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 595. Counter reading with index ungated (IPOS[1:0] = 00)

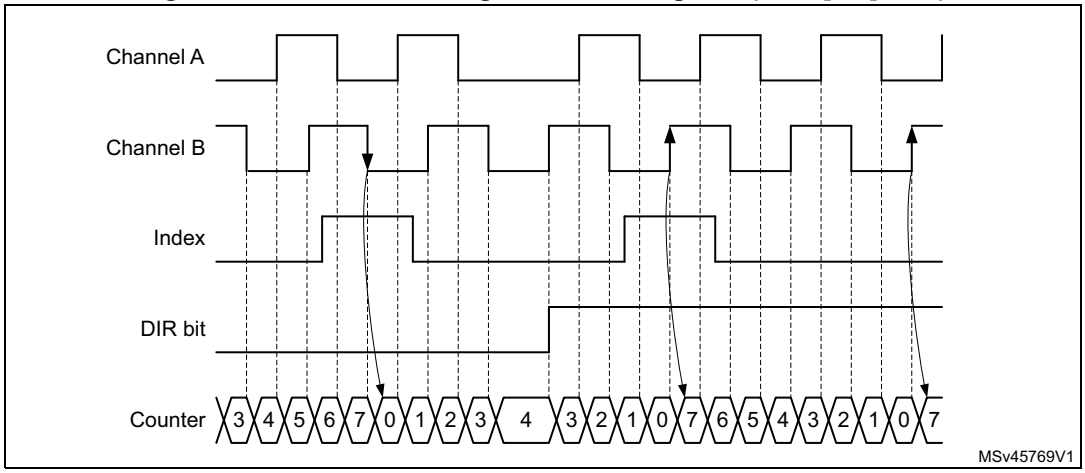


Figure 596. shows how the 'gated on A & B' mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.

Figure 596. Counter reading with index gated on channel A and B

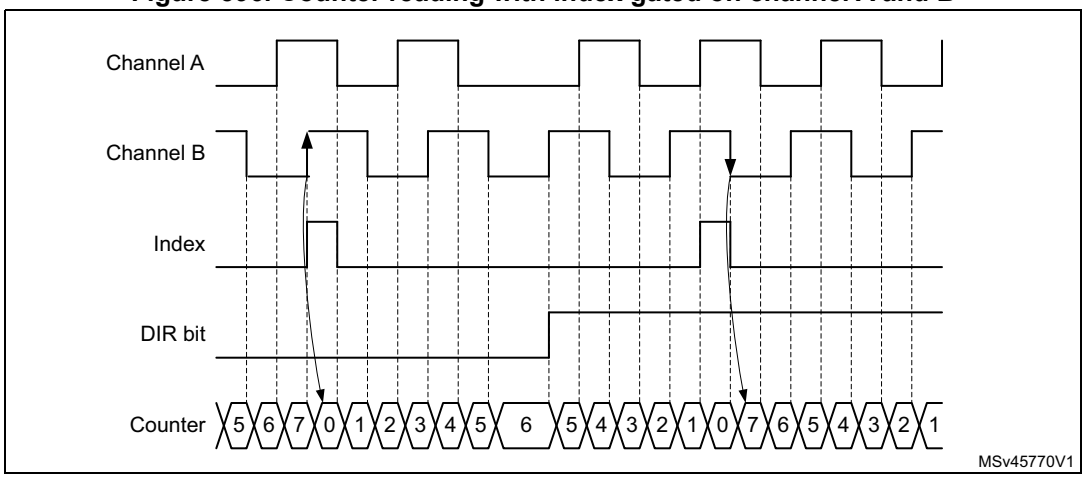


Figure 597 and *Figure 598* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 597. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

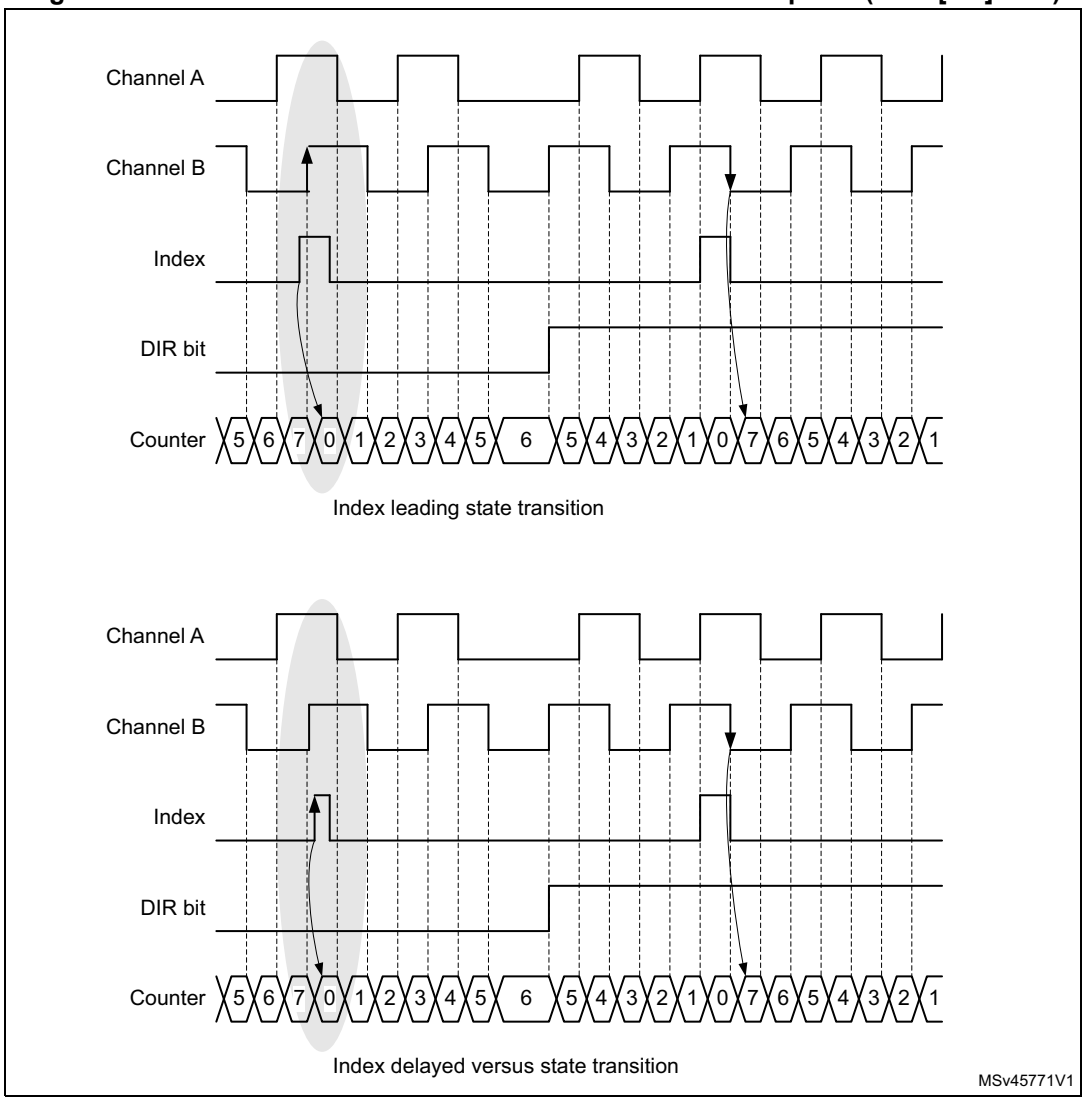


Figure 598. Counter reset Narrow index pulse (closer view, ARR = 0x07)

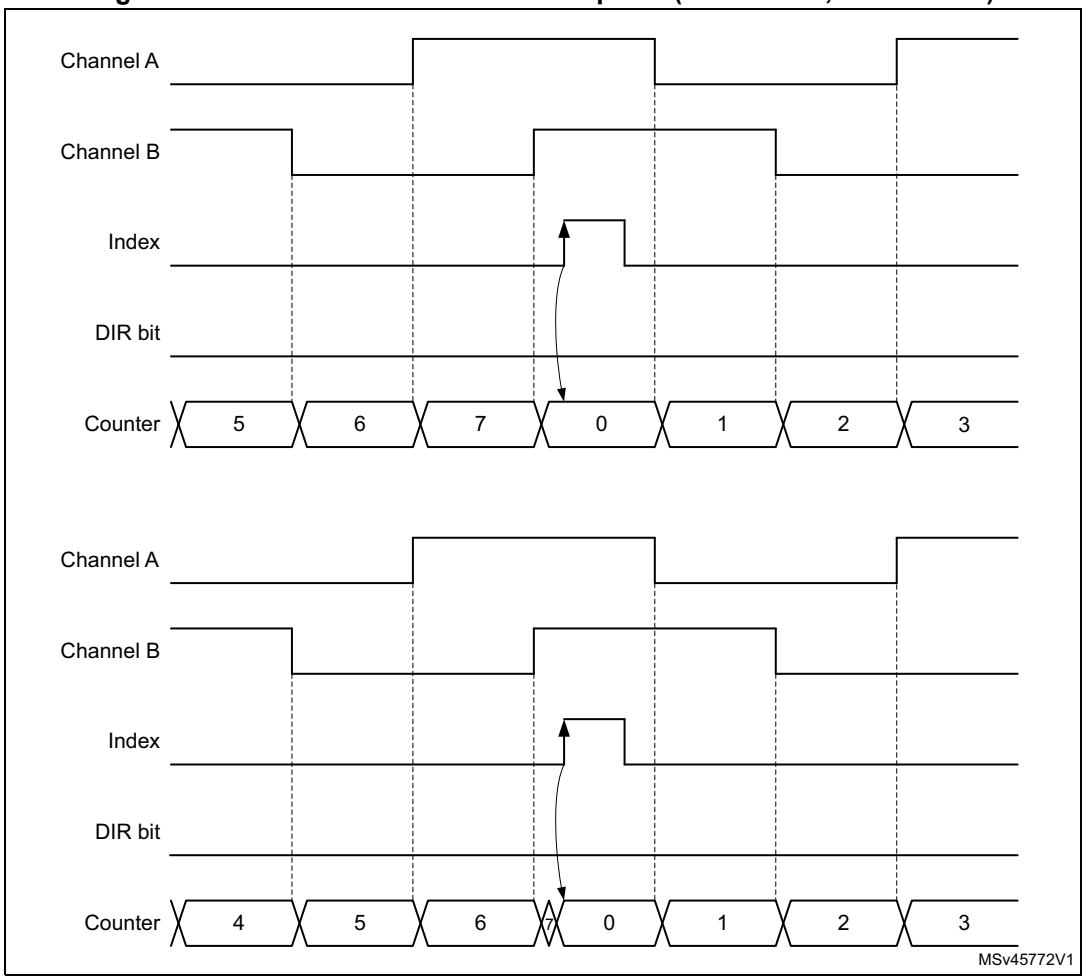
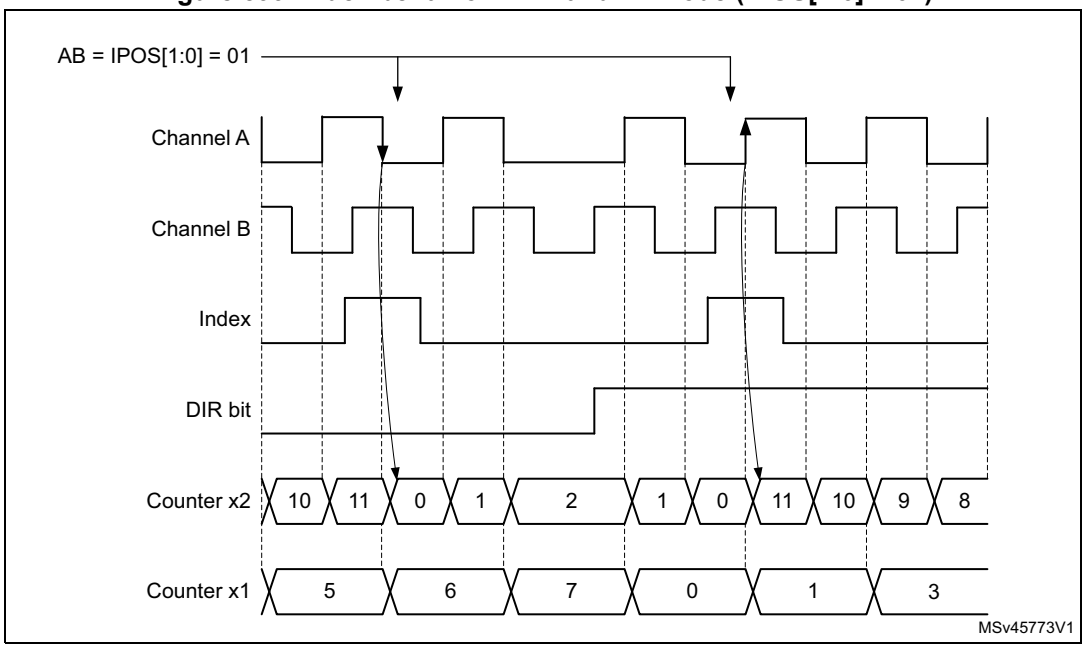


Figure 599 shows how the index is managed in x1 and x2 modes.

Figure 599. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

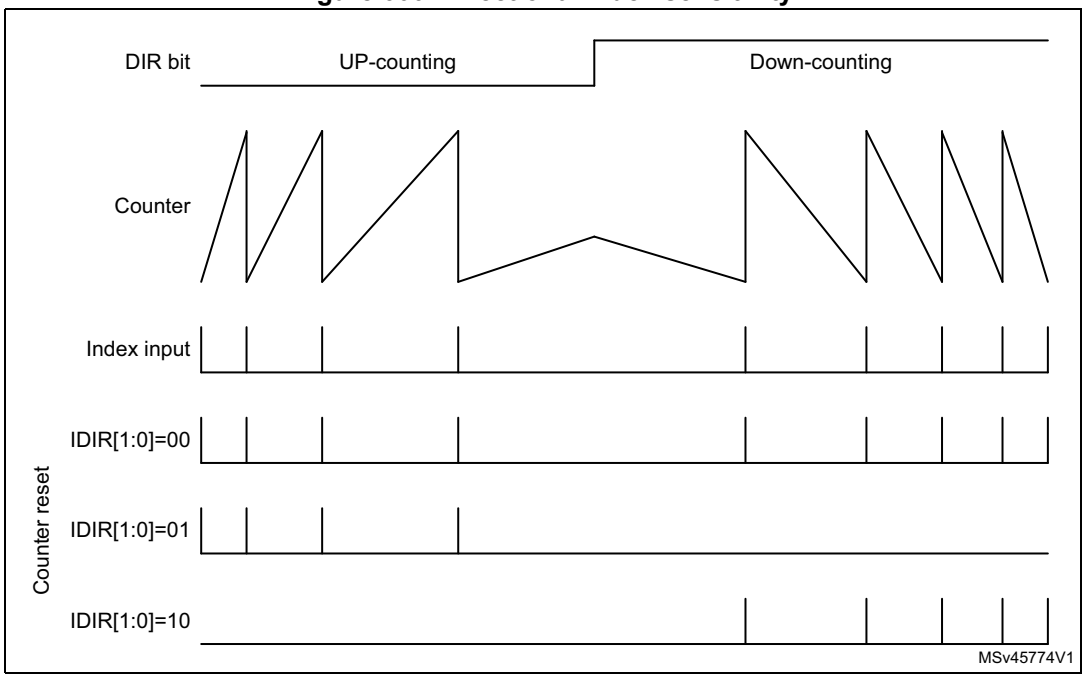


Directional index sensitivity

The IDIR[1:0] bitfield in the TIM_ECR register allows the index to be active only in a selected counting direction.

Figure 600 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

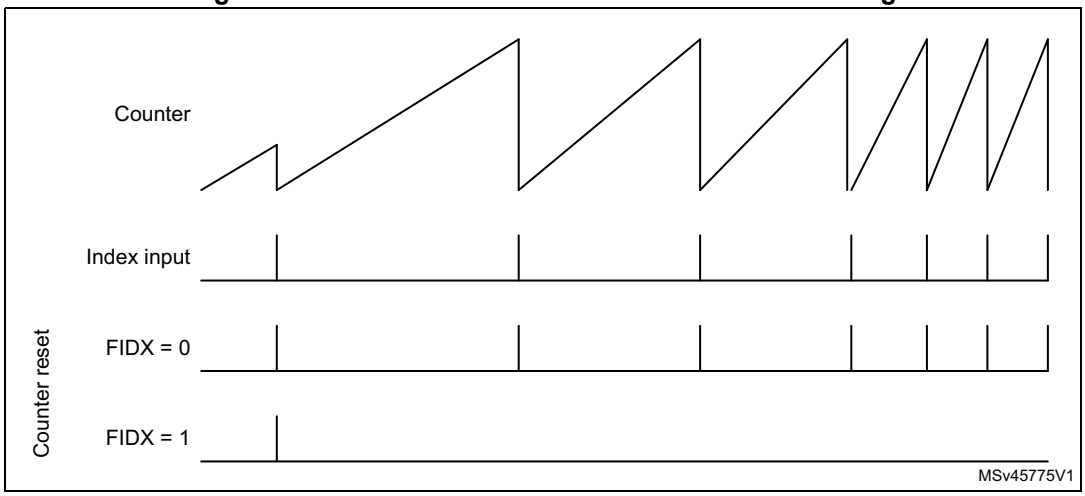
Figure 600. Directional index sensitivity



Special first index event management

The FIDX bit in the TIM_ECR register allows the index to be taken only once, as shown on *Figure 601*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

Figure 601. Counter reset as function of FIDX bit setting



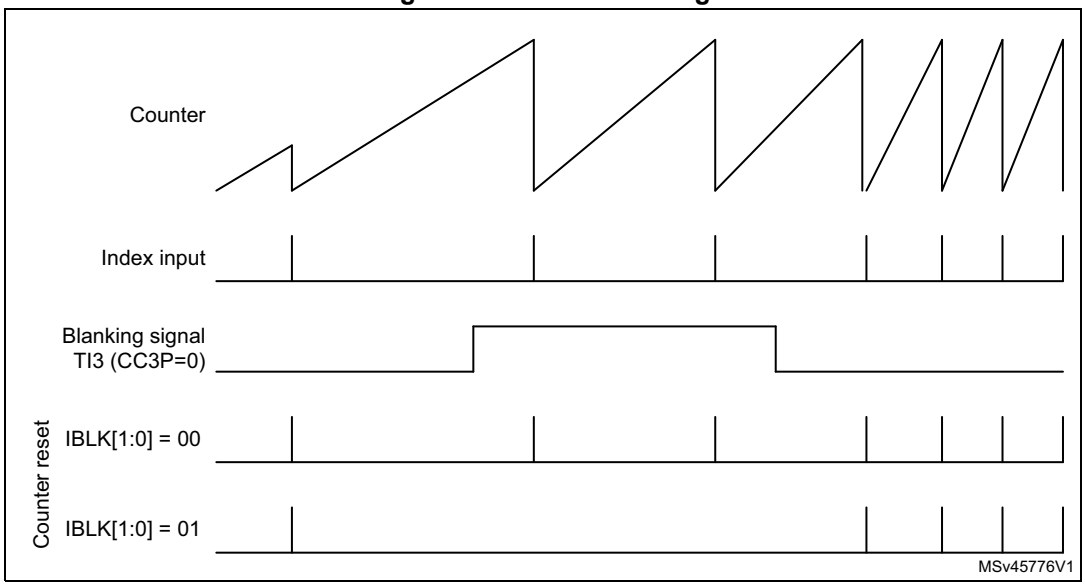
Index blanking

The index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on the *Figure 602* below.

This mode is enabled using the IBLK[1:0] bitfield in the TIM_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input

Figure 602. Index blanking



Index management in nonquadrature mode

Figure 603 and *Figure 604* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[4:0] bitfield is equal to 01010, 01011, 01100, 01101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level
- IPOS[0] = 1: Index is detected on clock high level

The IPOS[1] bit is not-significant.

Figure 603. Index behavior in clock + direction mode, IPOS[0] = 1

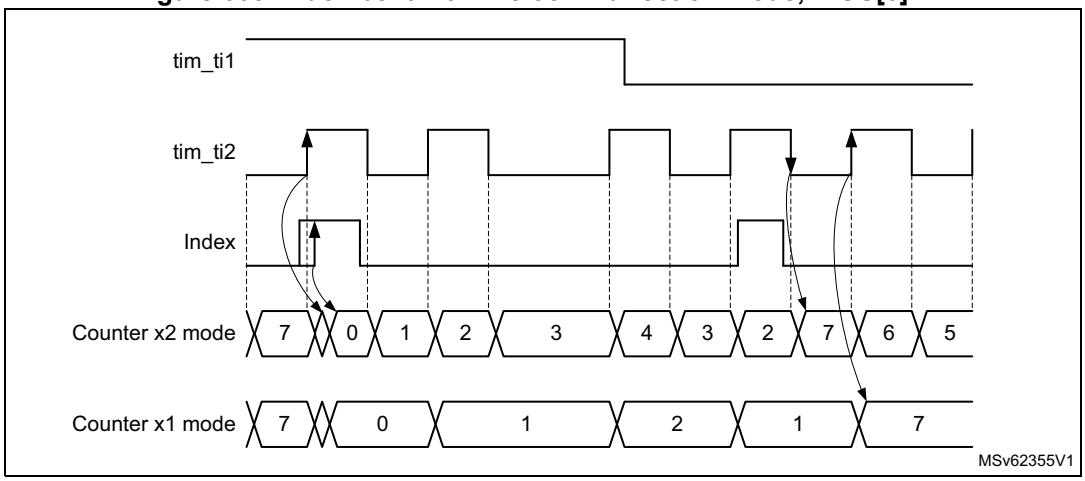
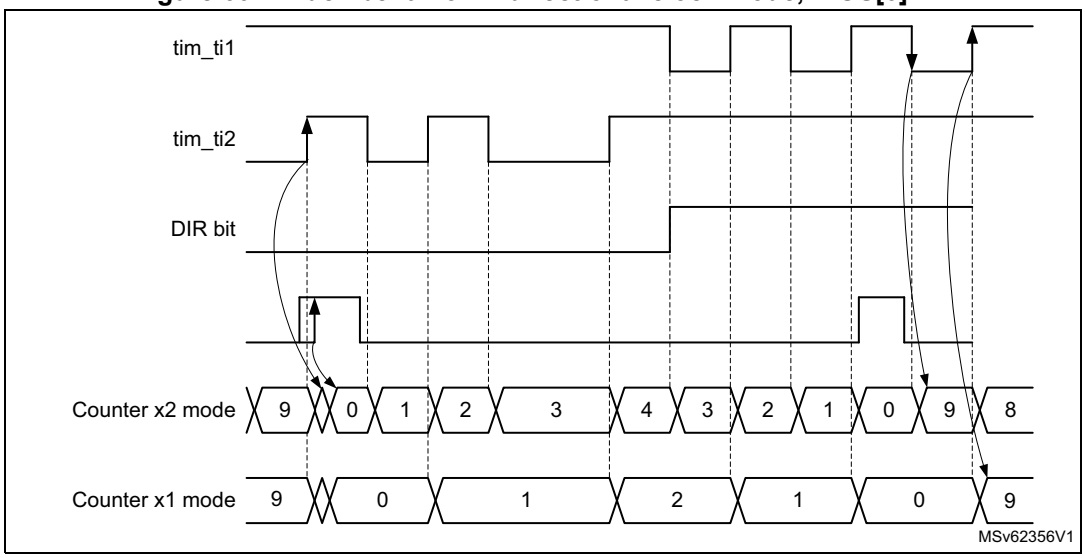


Figure 604. Index behavior in directional clock mode, IPOS[0] = 1

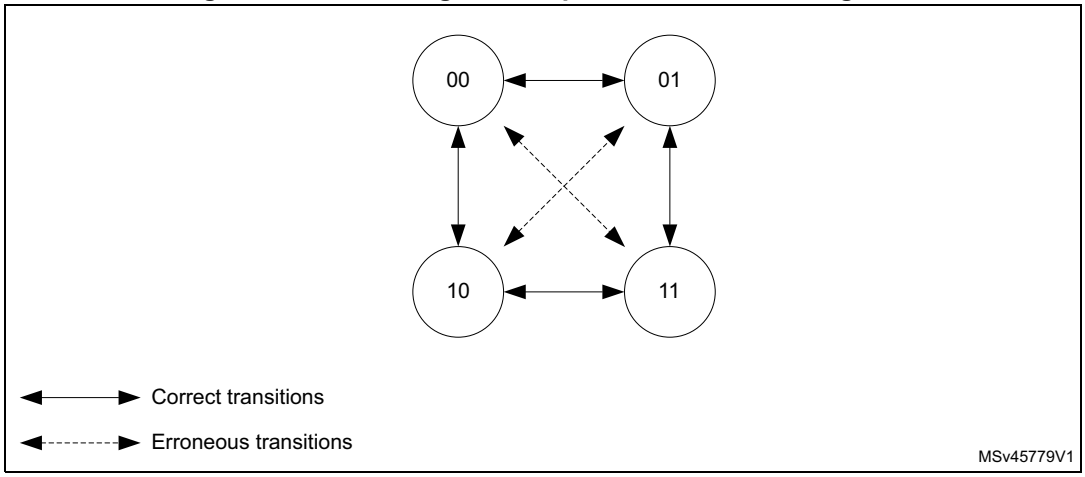


Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 605.*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIM_SR status

register. A transition error interrupt is generated if the TERRIE bit is set in the TIM_DIER register.

Figure 605. State diagram for quadrature encoded signals



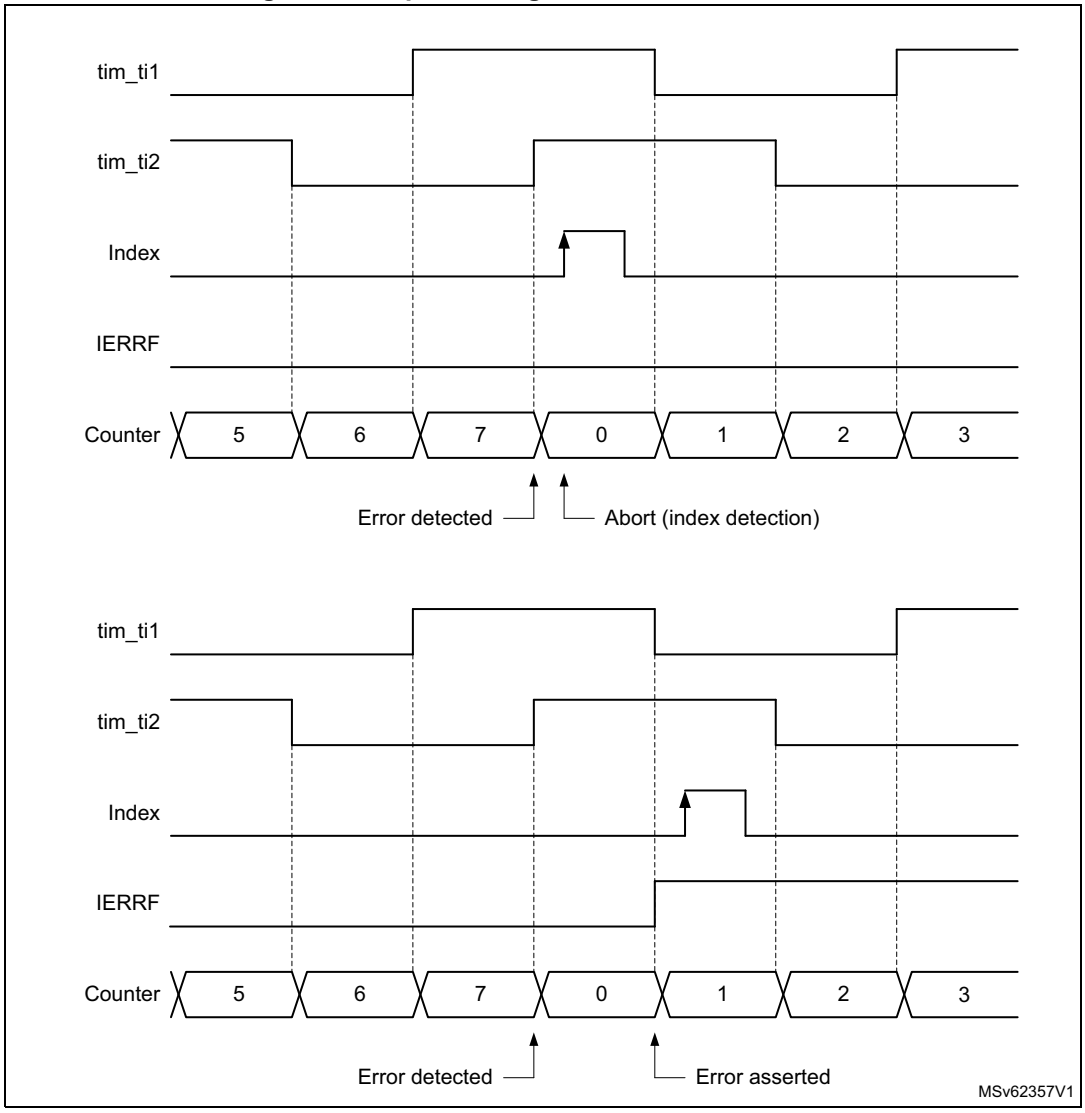
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIM_ARR to 0 or decremented from 0 to TIM_ARR value without any index event, this is reported as an index position error.

The overflow threshold is programmed using the TIM_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIM_ARR = 3999 + 1 = 4000.

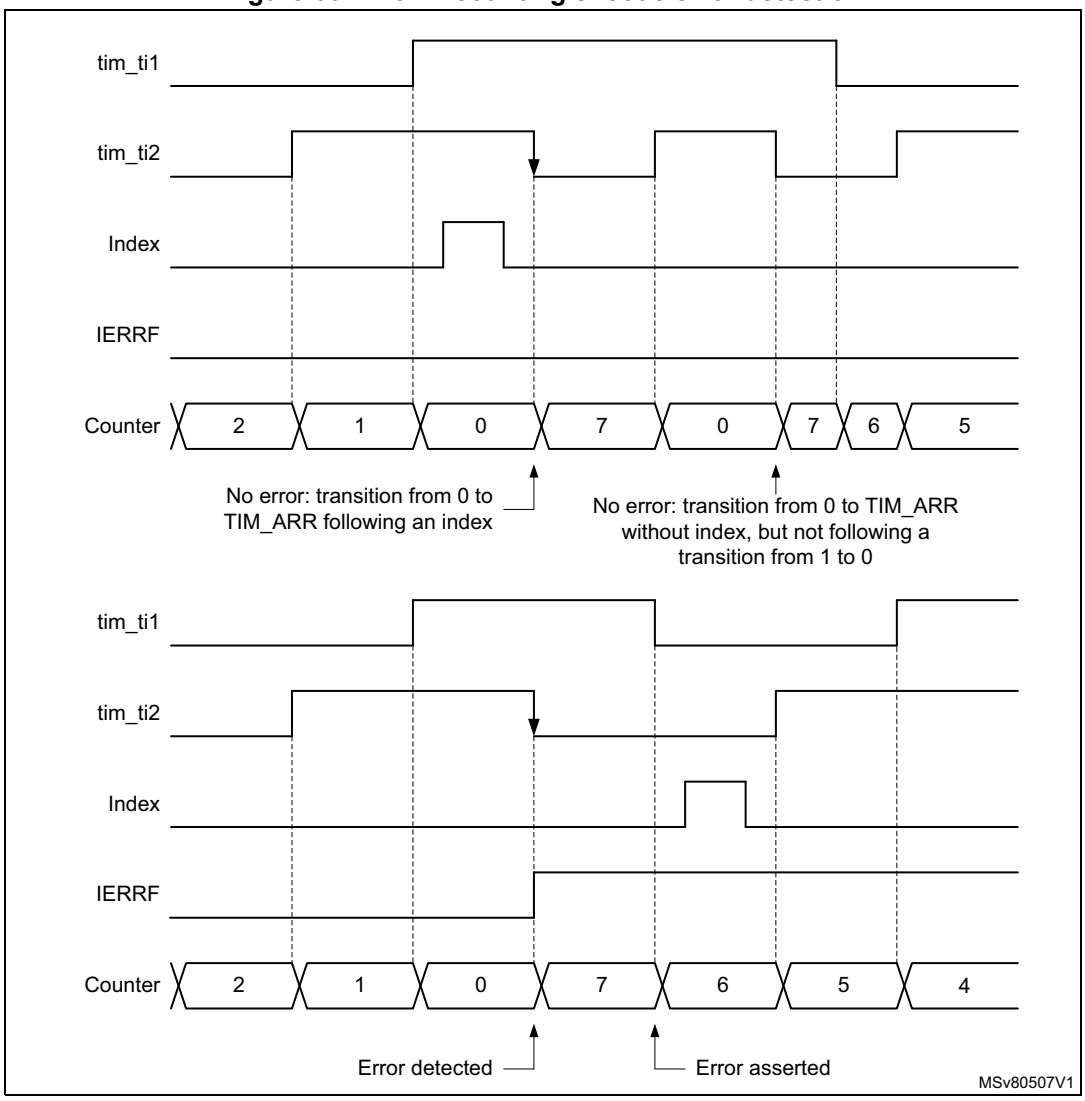
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is cope with narrow index pulses in gated A and B mode, as shown on *Figure 606*.

Figure 606. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 607*, to avoid any false error detection in case the encoder dithers between TIM_ARR and 0 immediately after the index detection.

Figure 607. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIM_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIM_DIER register.

Functional encoder interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIM_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIM_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIM_DIER register.
- Index event: the index event sets the IDXF interrupt flag in the TIM_SR status register. An index interrupt is generated if the IDXIE bit is set in the TIM_DIER register.

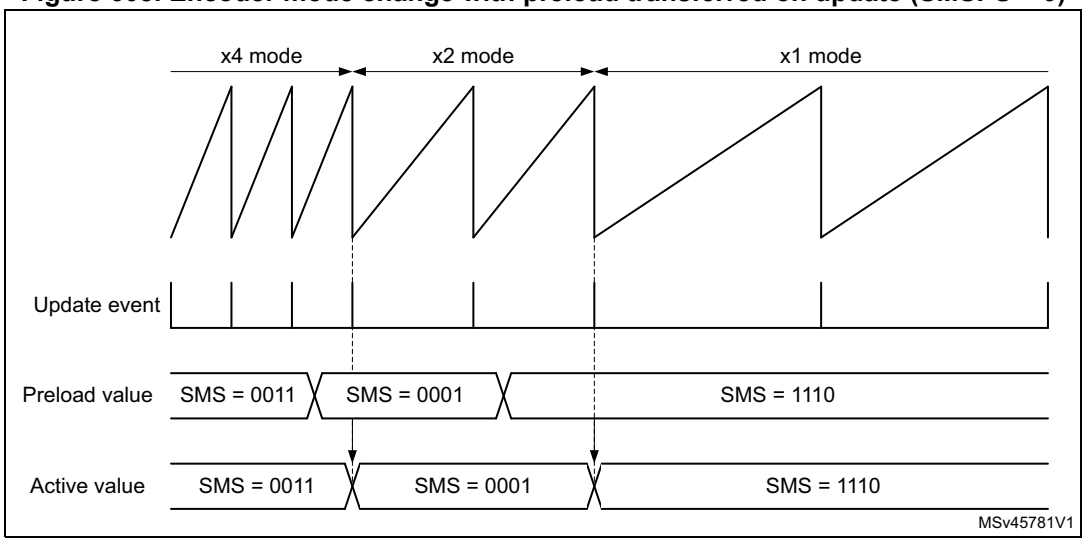
Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 608*.

For this purpose, the SMS[4:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIM_SMCR register. The trigger for the transfer from SMS[4:0] preload to active value can be selected with the SMSPS bit in the TIM_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 608. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIM_CR2 register.It is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

34.3.29 Encoder with built-in debouncer

This mode is intended to support noisy digital potentiometers and knobs, providing quadrature encoded signals with potentially long bouncing periods after each transition.

The operating principle consists of working on the first valid transition and discard any other until a second valid transition is detected on the alternate encoder signal.

The *Figure 609* below represents the x2 operating mode. It must be noticed that a transition on input 1 is ignored during direction reversal, if there's no edge on input 2.

The *Figure 610* below represents the x4 operating mode. In this case, a transition is always ignored during a direction reversal or when the encoder dithers between 2 positions.

These 2 modes are enabled by writing SMS[4:0] = 10000 or 10001 in TIM_SMCR register.

The encoder with built-in debouncer modes are not suitable for tracking a position or an angle in motor control applications. The transitions ignored during direction reversal will lead to a position/angle error that will build-up over time.

Figure 609. Encoder with built-in debouncer, x2 mode

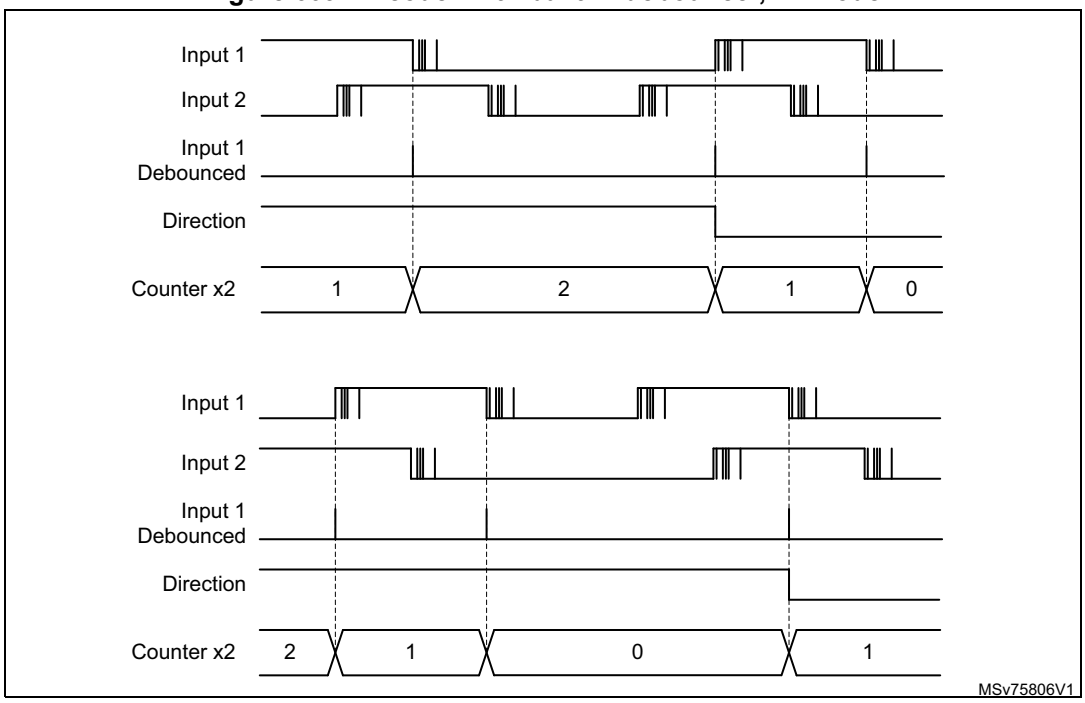
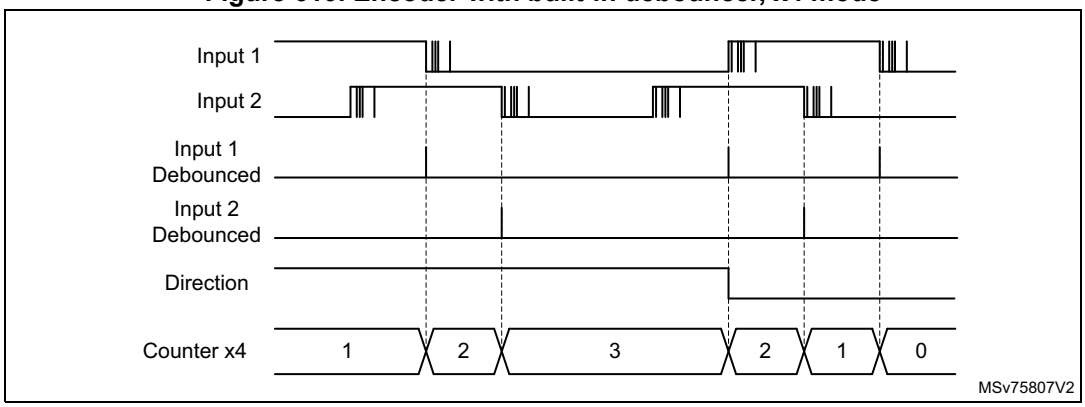


Figure 610. Encoder with built-in debouncer, x4 mode



34.3.30 Direction bit output

Its is possible to output a direction signal out of the timer, on the tim_oc3n and tim_oc4 output signals (copy of the DIR bit in the TIM_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bitfield to 1011 in the TIM_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

34.3.31 Clock drift measurement

It is possible to measure the drift between two clock sources using the directional clock encoder mode. This is useful for audio applications typically, when one need to compute the drift between two streams.

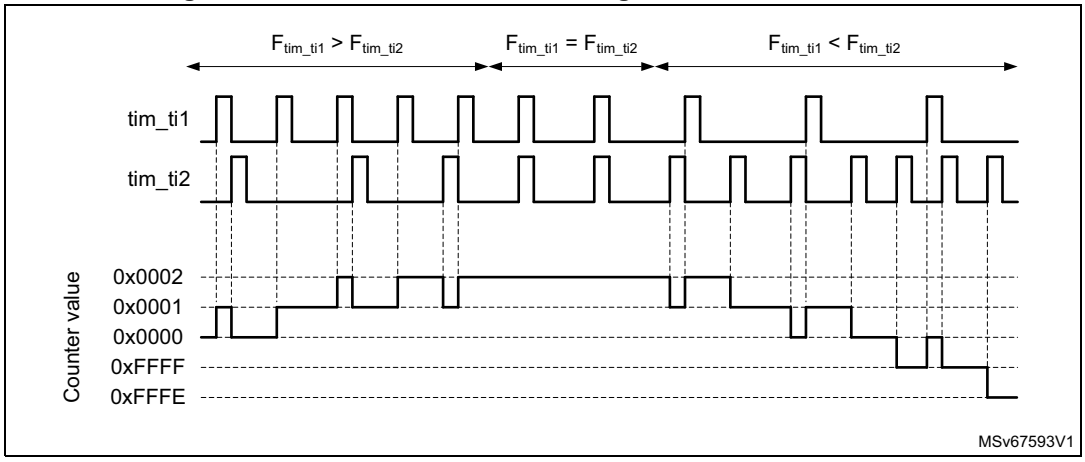
To do so, a signal representing the rate of a first audio stream must be connected to tim_ti1 and a signal representing the rate of a second audio stream must be connected to tim_ti2. The signals representing the audio rates are generally the SAIx_FS_A, SAIx_FS_B, SPIx_WS or spdifrx_frame_sync (this list of signals is provided as example and depends on the audio peripheral availability for a given product line, as detailed on the product datasheet).

In this case, the clocks are present on both tim_ti1 and tim2 simultaneously, as shown on *Figure 611*, and the counter value indicates the drift.

Let's define F_{tim_ti1} and F_{tim_ti2} the frequencies of the signals on tim_ti1 and tim_ti2. The counter value changes as following:

- If $F_{tim_ti1} > F_{tim_ti2}$ the counter value increases
- If $F_{tim_ti1} < F_{tim_ti2}$ the counter value decreases
- If $F_{tim_ti1} = F_{tim_ti2}$ the counter value dithers between two consecutive values, from one clock edge to the other. If the two clocks are perfectly synchronized (clock edges occuring simultaneously on the two inputs), the counter value does not change (as shown on *Figure 611* below).

Figure 611. Clock drift measure using directional clock mode



34.3.32 UIF bit remapping

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag UIF into the timer counter register's bit 31 (TIM_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

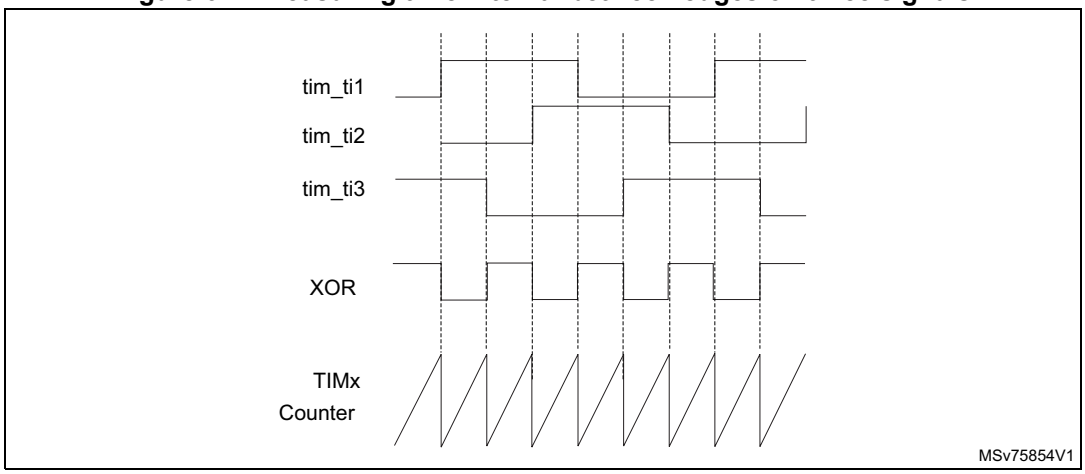
There is no latency between the UIF and UIFCPY flags assertion.

34.3.33 Timer input XOR function

The TI1S bit in the TIM_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per *Figure 612*.

Figure 612. Measuring time interval between edges on three signals



The XORPS control bit selects the position of the XOR gate, as shown on the *Figure 613* and *Figure 614* below, when the TI1S bit is set.

Figure 613. XOR gate connection with TI1S = 1 and XORPS = 0

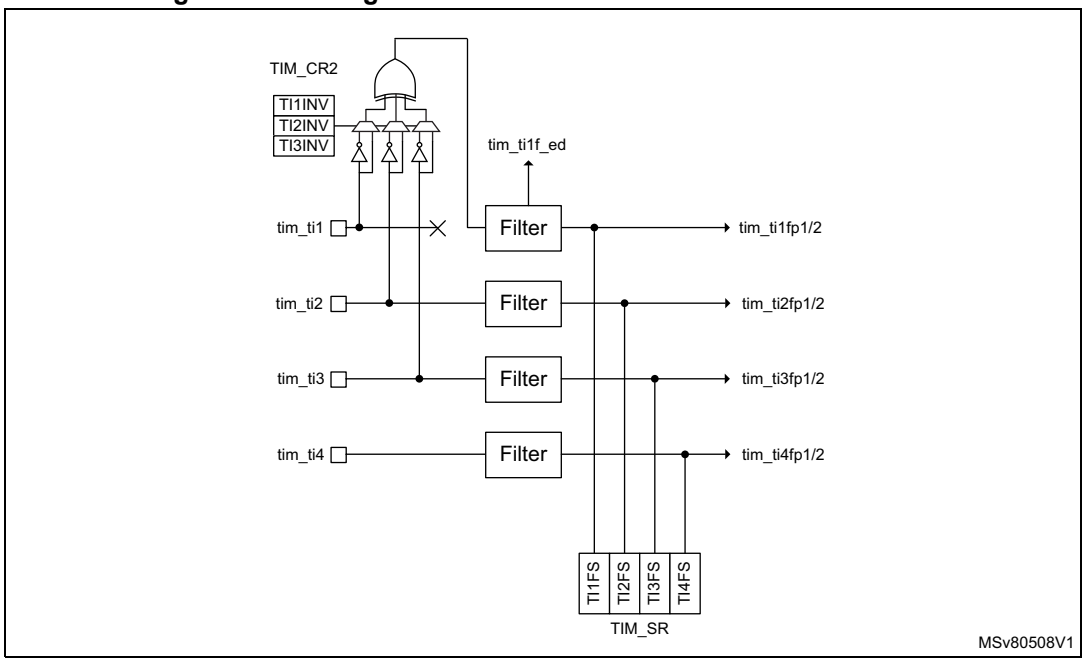
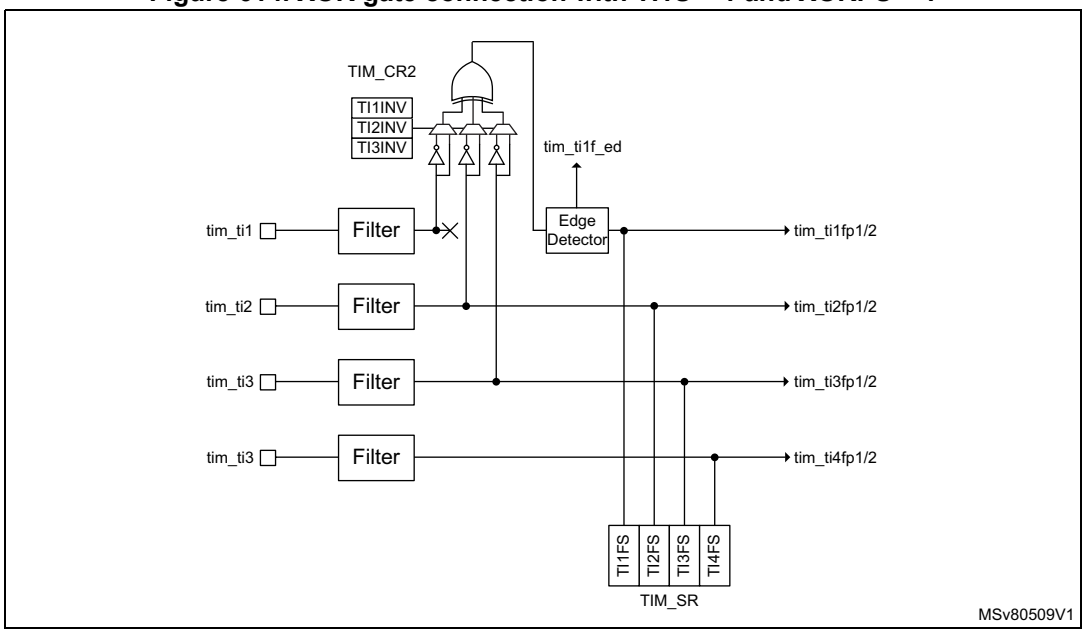


Figure 614. XOR gate connection with TI1S = 1 and XORPS = 1



Three control bits T1INV, T2INV and T3INV in the TIM_CR2 register can invert the tim_in1, tim_in2 and tim_in3 signals prior entering the XOR gate.

The status of the tim_in1, tim_in2, tim_in3 and tim_in4 signals can be software polled using the TI1FS, TI2FS, TI3FS and TI4FS bits in the TIM_SR register (tim_ti4 is not represented on figures above and do not enter into the XOR gate).

34.3.34 Interfacing with Hall sensors

This is done using the advanced-control timers to generate PWM signals to drive the motor and another timer TIMx referred to as “interfacing timer” in *Figure 615*. The “interfacing timer” captures the three timer input pins (tim_ti1, tim_ti2 and tim_ti3) connected through a XOR to the tim_ti1 input channel (selected by setting the TI1S bit in the TIM_CR2 register, with XORPS bit = 0).

The slave mode controller is configured in reset mode; the slave input is tim_ti1f_ed. Thus, each time one of the three inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is tim_trc (See *Figure 541*). The captured value, which corresponds to the time elapsed between two changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (by triggering a COM event). The advanced-control timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer through the tim_trgo output.

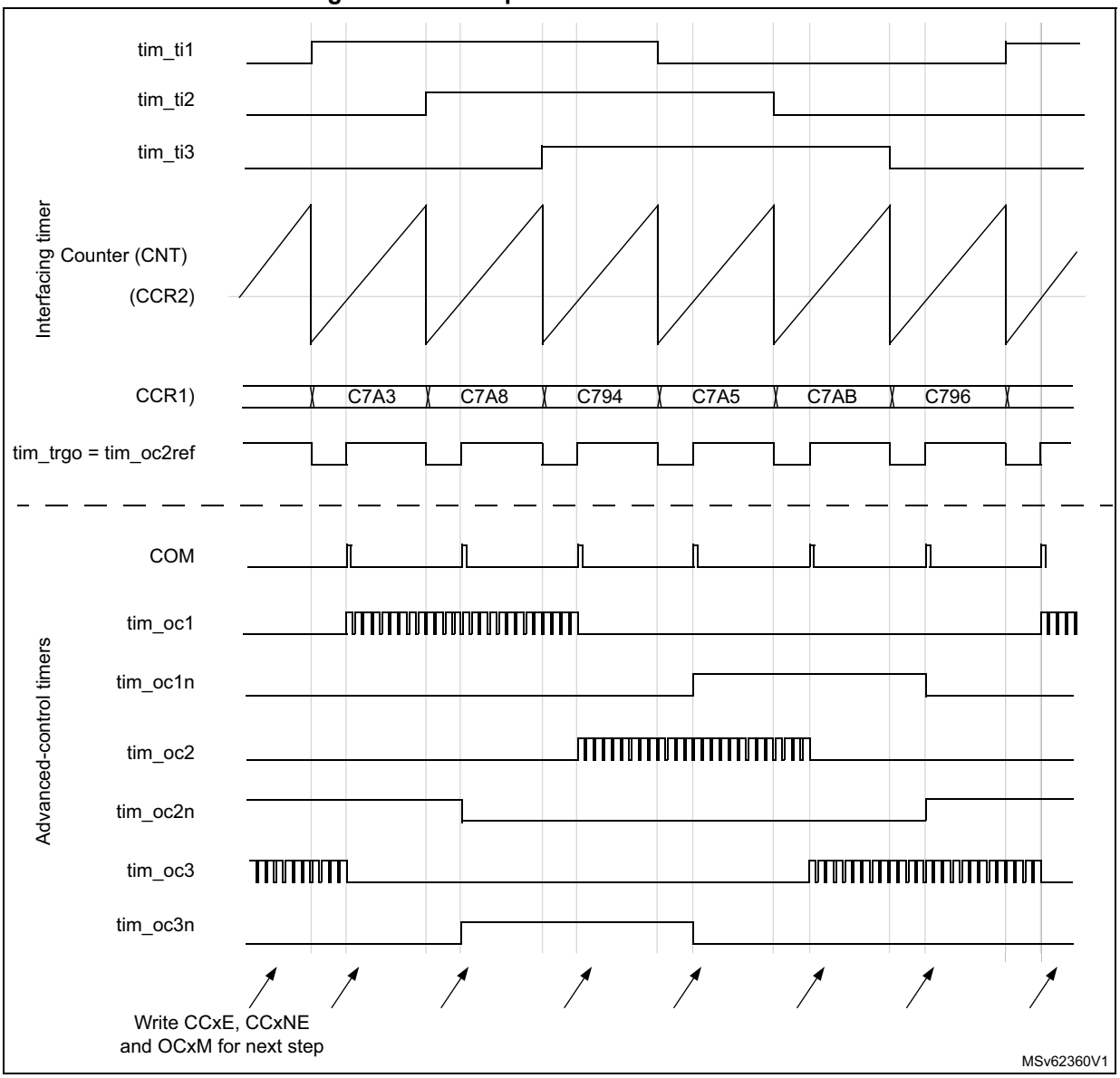
In this example the user wants to change the PWM configuration of the advanced-control timer after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure three timer inputs ORed to the tim_ti1 input channel by writing the TI1S bit in the TIM_CR2 register to 1.
- Program the time base: write the TIM_ARR to the max value (the counter must be cleared by the tim_ti1 change. Set the prescaler to get a maximum counter period longer than the time between two changes on the sensors.
- Program the channel 1 in capture mode (tim_trc selected): write the CC1S bits in the TIM_CCMR1 register to 01. The digital filter can also be programmed if needed.
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to 111 and the CC2S bits to 00 in the TIM_CCMR1 register.
- Select tim_oc2ref as trigger output on tim_trgo: write the MMS bits in the TIM_CR2 register to 101.

In the advanced-control timer, the right tim_itrx input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC = 1 in the TIM_CR2 register) and the COM event is controlled by the trigger input (CCUS = 1 in the TIM_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of tim_oc2ref).

Figure 615 describes this example.

Figure 615. Example of Hall sensor interface



34.3.35 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 35.4.27: Timer synchronization* for details. They can be synchronized in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIM_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIM_ARR, TIM_CCRx) are updated.

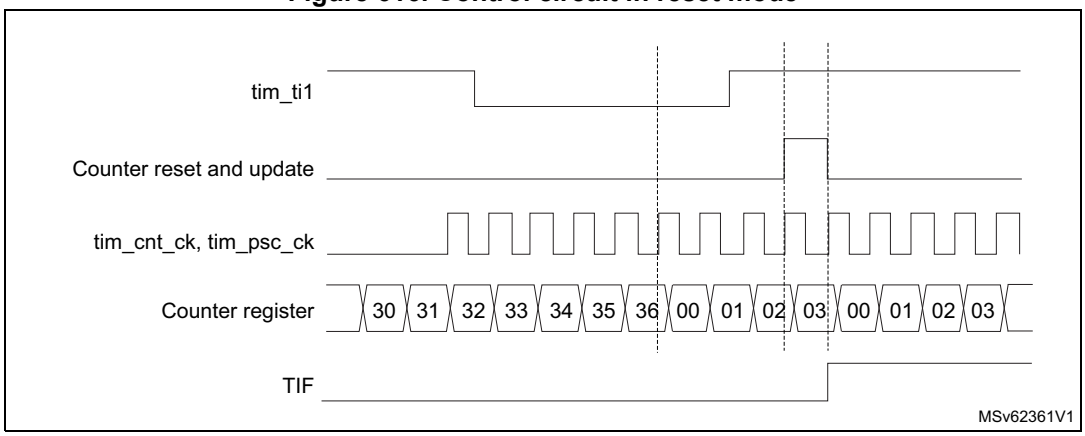
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

- Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIM_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS = 100 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
- Start the counter by writing CEN = 1 in the TIM_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIM_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIM_DIER register).

The following figure shows this behavior when the autoreload register TIM_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 616. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

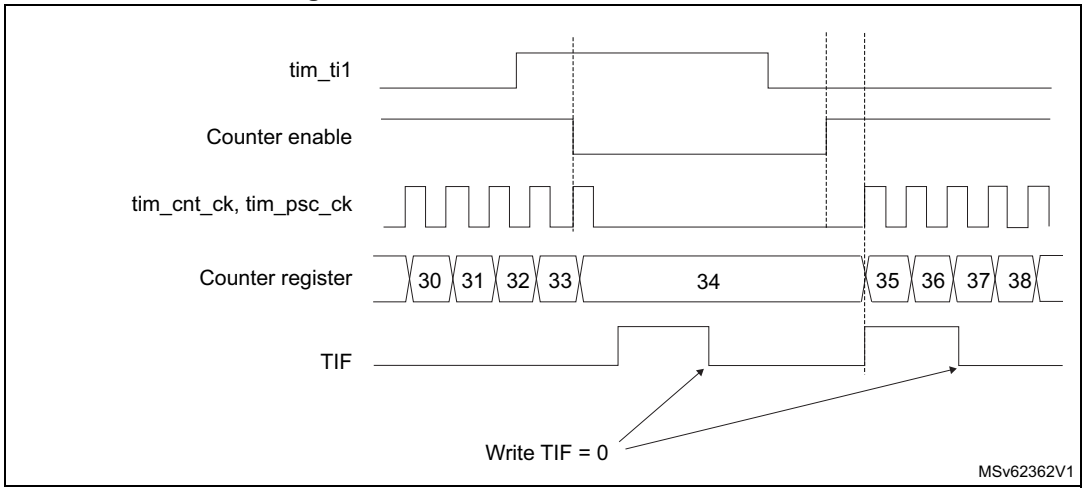
In the following example, the upcounter counts only when tim_ti1 input is low:

- Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIM_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS = 101 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
- Enable the counter by writing CEN = 1 in the TIM_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIM_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 617. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

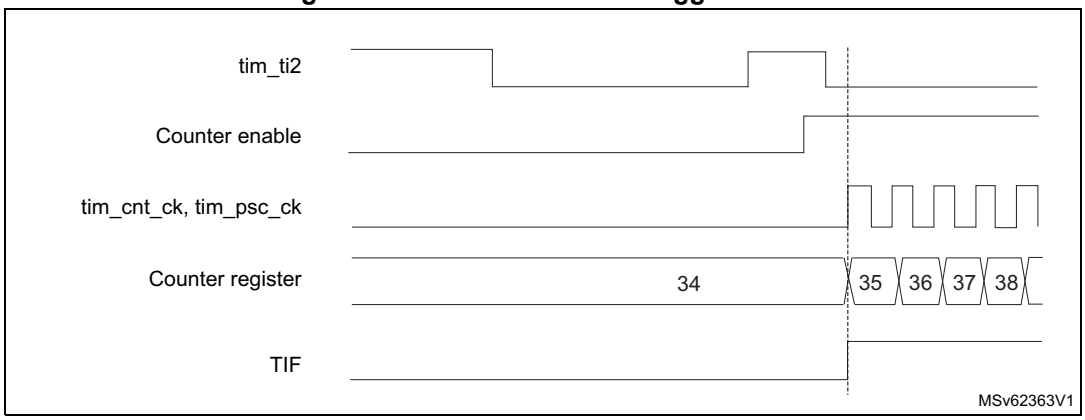
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

- Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIM_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS = 110 in TIM_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIM_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 618. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for One-pulse mode.

Slave mode: Combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIM_SMCR register.

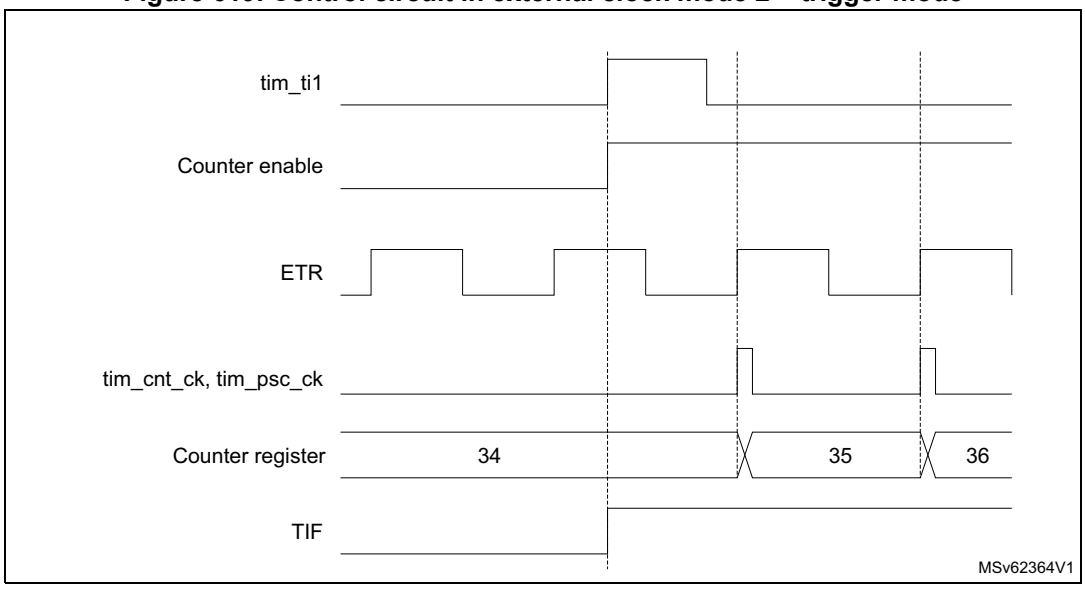
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIM_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS = 00: prescaler disabled
   - ETP = 0: detection of rising edges on tim_etr_in and ECE = 1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in TIM_CCMR1 register to select only the input capture source
   - CC1P = 0 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS = 110 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 619. Control circuit in external clock mode 2 + trigger mode



Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

34.3.36 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

Refer to MMS2[4:0] bitfield description in TIM_CR2 for the complete list of triggering options.

The triggers are issued on the tim_trgo2 internal line which is redirected to the ADC. There is a total of 24 possible events, which can be selected using the MMS2[4:0] bits in the TIM_CR2 register.

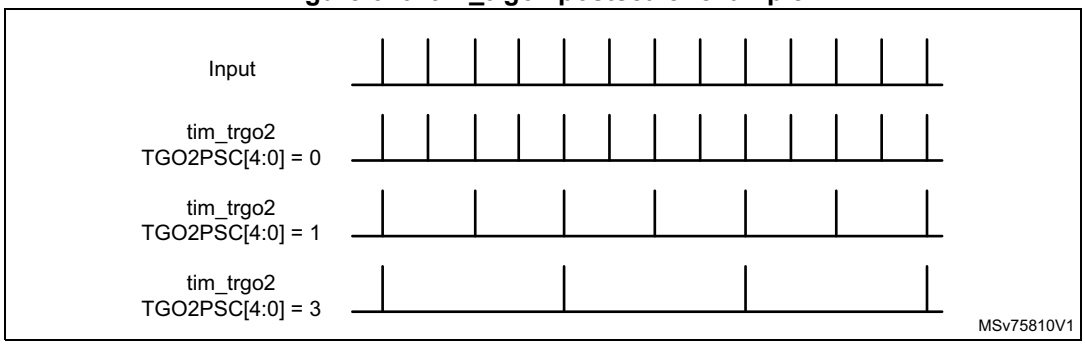
An example of an application for 3-phase motor drives is given in *Figure 559*.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Note: The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

It is possible to reduce the ADC triggering rate with the tim_tro2 post scaler. This is done by periodically discarding tim_trgo2 events, as per the value programmed in TGO2PSC[4:0] bitfield in TIM_CR1, as shown on *Figure 620* below.

Figure 620. tim_trgo2 postscaler example



The post scaler is able to operate with:

– pulses as shown on *Figure 620* above, when selecting for instance MMS2[4:0] = 10001 to have a trigger on tim_oc7refc rising rising or falling edges

– the tim_ocxrefc signal itself (ie with user-defined duty cycle), when selecting for instance MMS2[4:0] = 10000 to have a trigger on tim_oc7refc

34.3.37 ADC synchronization

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIM_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- when the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum
- when the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source).

1. The counter period must be a multiple of the ADC clock period:

$$(\text{TIM_PSC} + 1) \times (\text{TIM_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period:

$$(\text{TIM_PSC} + 1) \times \text{TIM_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements are met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock.
2. Configure the timer and set the ADSYNC bit.
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits).
4. Start the timer (with the CEN counter enable bit).

34.3.38 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to reprogram part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIM_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIM_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIM_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers (in words).

The DBA[4:0] bits in the TIM_DCR register define the DMA base address for DMA transfers (when read/write access are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register:

Example:

00000: TIM_CR1

00001: TIM_CR2

00010: TIM_SMCR

The DBSS[3:0] bits in the TIM_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 34.6.34: TIM DMA control register (TIM_DCR)* for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address.
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (see note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bitfields as follows: DBL = 2, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx.
5. Enable the DMA channel.

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

34.3.39 TIM1/TIM8/TIM20 DMA requests

The TIM1/TIM8/TIM20 can generate a DMA request, as shown in the table below.

Table 387. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_cc3_dma	Capture/compare 3	CC3DE

Table 387. DMA request (continued)

DMA request signal	DMA request	Enable control bit
tim_cc4_dma	Capture/compare 4	CC4DE
tim_com_dma	Commutation (COM)	COMDE
tim_trgi_dma	Trigger	TDE

34.3.40 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter can either continue to work normally or stop, depending on DBG_TIM_STOP configuration bit in DBG module.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to section Debug support (DBG).

34.4 TIM1/TIM8/TIM20 low-power modes

Table 388. Effect of low-power modes on TIM1/TIM8/TIM20

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

34.5 TIM1/TIM8/TIM20 interrupts

The TIM1/TIM8/TIM20 can generate multiple interrupts, as shown in *Table 389*.

Table 389. Interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UPD		Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC		Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
		Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
		Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
		Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRGI_COM_DIR_IDX	TIM_COM	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	TIM_TRGI	Trigger	TIF	TIE	write 0 in TIF	Yes	No
	TIM_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	TIM_DIR	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_BRK_TERR_IERR	TIM_BRK	Break	BIF	BIE	write 0 in BIF	Yes	No
		Break2	B2IF		write 0 in B2IF	Yes	No
		System Break	SBIF		write 0 in SBIF	Yes	No
	TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
	TIM_TERR	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

34.6 TIM1/TIM8/TIM20 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

34.6.1 TIM control register 1 (TIM_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	TGO2PSC[4:0]				
				rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CKD2[3:0]**: Clock division 2

This bit field indicates the division between sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_etr_in, tim_tix).

0000: No Division
0001: $t_{DTS2} = 4*t_{DTS}$
0010: $t_{DTS2} = 16*t_{DTS}$
0011: $t_{DTS2} = 64*t_{DTS}$
0100: $t_{DTS2} = 256*t_{DTS}$
0101: $t_{DTS2} = 1024*t_{DTS}$
0110: $t_{DTS2} = 4096*t_{DTS}$
0111: $t_{DTS2} = 16384*t_{DTS}$
1000: $t_{DTS2} = 65536*t_{DTS}$
1001: $t_{DTS2} = 262144*t_{DTS}$
Others: Reserved, do not program this value

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **TGO2PSC[4:0]**: Trigger output 2 postcaler

The postscaler discards periodically events sent on the tim_trgo2 output, to reduce the ADC triggering rate.
The number of events on tim_trgo2 is divided by (TGO2PSC[4:0] + 1).
When TGO2PSC[4:0] = 0x0, the tim_trgo2 event rate is not affected.
With TGO2PSC[4:0] = 0x1, one event out of 2 is sent on tim_trgo2, with TGO2PSC[4:0] = 0x2, one event out of 3 is sent on tim_trgo2, up to 1 event out of 31 when TGO2PSC[4:0] = 0x1F,...

Note: The postscaler is only applicable when tim_trgo2 transfers a pulse (reset, update, compare pulse). When the tim_trgo2 outputs a level-based information (compare signal or enable bit for gated mode), the postscaler must be set set to 0

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_etr_in, tim_tix),
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: $t_{DTS} = 8*t_{tim_ker_ck}$

Bit 7 **ARPE**: Autoreload preload enable
0: TIM_ARR register is not buffered
1: TIM_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection
00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set both when the counter is counting up or down.
Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter
Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

34.6.2 TIM control register 2 (TIM_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TI3INV	TI2INV	TI1INV	ADSYNC	XORPS	Res.	MMS[3]	MMS2[4:0]					OIS7	OIS6	Res.	OIS5
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit 31 **TI3INV**: tim_ti3 signal inversion on XOR gate input

0: tim_ti3 non-inverted

1: tim_ti3 inverted

Bit 30 **TI2INV**: tim_ti2 signal inversion on XOR gate input

0: tim_ti2 non-inverted

1: tim_ti2 inverted

Bit 29 **TI1INV**: tim_ti1 signal inversion on XOR gate input

0: tim_ti1 non-inverted

1: tim_ti1 inverted

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 34.3.37* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bit 27 **XORPS**: XOR gate position selection

This bit defines the position of the XOR gate with respect to input filters.

0: XOR gate placed before TI1 filter

1: XOR gate placed after TI1, TI2 and TI3 filters

Bit 26 Reserved, must be kept at reset value.

Bits 24:20 **MMS2[4:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (tim_trgo2) to be selected. The combination is as follows:

00000:**Reset** - the UG bit from the TIM_EGR register is used as trigger output (tim_trgo2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on tim_trgo2 is delayed compared to the actual reset.

00001:**Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo2, except if the Master/Slave mode is selected (see the MSM bit description in TIM_SMCR register).

00010:**Update** - the update event is selected as trigger output (tim_trgo2). For instance, a master timer can then be used as a prescaler for a slave timer.

00011:**Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (tim_trgo2).

00100:**Compare** - tim_oc1refc signal is used as trigger output (tim_trgo2)

00101:**Compare** - tim_oc2refc signal is used as trigger output (tim_trgo2)

00110:**Compare** - tim_oc3refc signal is used as trigger output (tim_trgo2)

00111:**Compare** - tim_oc4refc signal is used as trigger output (tim_trgo2)

01000:**Compare** - tim_oc5refc signal is used as trigger output (tim_trgo2)

01001:**Compare** - tim_oc6refc signal is used as trigger output (tim_trgo2)

01010:**Compare Pulse** - tim_oc4refc rising or falling edges generate pulses on tim_trgo2

01011:**Compare pulse** - tim_oc6refc rising or falling edges generate pulses on tim_trgo2

01100:**Compare pulse** - tim_oc4refc or tim_oc6refc rising edges generate pulses on tim_trgo2

01101:**Compare pulse** - tim_oc4refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

01110:**Compare pulse** - tim_oc5refc or tim_oc6refc rising edges generate pulses on tim_trgo2

01111:**Compare pulse** - tim_oc5refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

10000: **Compare** - tim_oc7refc signal is used as trigger output (tim_trgo2)

10001: **Compare Pulse** - tim_oc7refc rising or falling edges generate pulses on tim_trgo2

10010: **Compare Pulse** tim_oc4refc or tim_oc7efc rising edges generate pulses on tim_trgo2

10011: **Compare Pulse** - tim_oc4refc rising or tim_oc7refc falling edges generate pulses on tim_trgo2

10100: **Compare Pulse** -tim_oc5refc or tim_oc7efc rising edges generate pulses on tim_trgo2

10101: **Compare Pulse** - tim_oc5refc rising or tim_oc7refc falling edges generate pulses on tim_trgo2

10110: **Compare Pulse** - tim_oc6refc or tim_oc7efc rising edges generate pulses on tim_trgo2

10111: **Compare Pulse** - tim_oc6refc rising or tim_oc7refc falling edges generate pulses on tim_trgo2

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19 **OIS7**: Output idle state 7 (tim_oc7 output)

Refer to OIS1 bit

Bit 18 **OIS6**: Output idle state 6 (tim_oc6 output)
Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output idle state 5 (tim_oc5 output)
Refer to OIS1 bit

Bit 15 **OIS4N**: Output idle state 4 (tim_oc4n output)
Refer to OIS1N bit

Bit 14 **OIS4**: Output idle state 4 (tim_oc4 output)
Refer to OIS1 bit

Bit 13 **OIS3N**: Output idle state 3 (tim_oc3n output)
Refer to OIS1N bit

Bit 12 **OIS3**: Output idle state 3 (tim_oc3n output)
Refer to OIS1 bit

Bit 11 **OIS2N**: Output idle state 2 (tim_oc2n output)
Refer to OIS1N bit

Bit 10 **OIS2**: Output idle state 2 (tim_oc2 output)
Refer to OIS1 bit

Bit 9 **OIS1N**: Output idle state 1 (tim_oc1n output)
0: tim_oc1n = 0 after a dead-time when MOE = 0
1: tim_oc1n = 1 after a dead-time when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bit 8 **OIS1**: Output idle state 1 (tim_oc1 output)
0: tim_oc1 = 0 (after a dead-time) when MOE = 0
1: tim_oc1 = 1 (after a dead-time) when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bit 7 **TI1S**: tim_ti1 selection
This bit enables the insertion of a XOR gate to combine tim_ti1_in[15:0], tim_ti2_in[15:0], and tim_ti3_in[15:0] inputs into tim_ti1.
The position of the XOR gate, with respect to input filters, is defined with the XORPS bit.
0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input
1: tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input

Bits 25, 6:4 **MMS[3:0]**: Master mode selection

These bits select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIM_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

Other codes reserved

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE, CCxP, CCxNP and OCxM bits are not preloaded

1: CCxE, CCxNE, CCxP, CCxNP and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

34.6.3 TIM slave mode control register (TIM_SMCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SETPS[3:0]				SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	SMS[4:3]	
		rw	rw	rw	rw	rw	rw			rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:26 **SETPS[3:0]**: Synchronous external trigger prescaler

The linear synchronous prescaler divides the frequency of the tim_etrf signal, by a factor (SETPS[3:0] + 1).

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[4:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded

0: SMS[4:0] bitfield is not preloaded

1: SMS[4:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **TS[4:3]**: Trigger selection - bit 4:3

Refer to TS[2:0] description - bits 6:4

Bits 19:18 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or $\overline{\text{tim_etr_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge.

1: tim_etr_in is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS = 111 and TS = 00111).

It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).

If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF
01: tim_etr_in frequency divided by 2
10: tim_etr_in frequency divided by 4
11: tim_etr_in frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bitfield then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS2}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS2}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS2}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS2}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS2}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS2}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS2}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS2}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS2}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS2}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS2}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS2}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS2}/32$, N = 8

Bit 7 **MSM:** Master/slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]:** Trigger selection

This bitfield is combined with TS[4:3] bits.

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal trigger 0 (tim_itr0)
00001: Internal trigger 1 (tim_itr1)
00010: Internal trigger 2 (tim_itr2)
00011: Internal trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: External trigger input (tim_etrf)
01000: Internal trigger 4 (tim_itr4)
01001: Internal trigger 5 (tim_itr5)
01010: Internal trigger 6 (tim_itr6)
01011: Internal trigger 7 (tim_itr7)
01100: Internal trigger 8 (tim_itr8)
01101: Internal trigger 9 (tim_itr9)
01110: Internal trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
10100: Internal trigger 16 (tim_itr16)
10101: Internal trigger 17 (tim_itr17)
10110: Internal trigger 18 (tim_itr18)
10111: Internal trigger 19 (tim_itr19)
11000: Internal trigger 20 (tim_itr20)
11001: Internal trigger 21 (tim_itr21)
11010: Internal trigger 22 (tim_itr22)
11011: Internal trigger 23 (tim_itr23)
11100: Internal trigger 24 (tim_itr24)
11101: Internal trigger 25 (tim_itr25)
11110: Internal trigger 26 (tim_itr26)
11111: Internal trigger 27 (tim_itr27)

See *Section 34.3.2: TIM1/TIM8/TIM20 pins and internal signals* for more details on tim_itrx meaning for each Timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS:** OCREF clear selection

This bit is used to select the OCREF clear source.

0: tim_ocref_clr_int is connected to the tim_ocref_clr input
1: tim_ocref_clr_int is connected to tim_etrf

Bits 17:16, 2:0 **SMS[4:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIM_SMCR for tim_etr_in and CCxP/CCxNP bits in TIM_CCER register for tim_ti1fp1 and tim_ti2fp2).

00000:Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

00001:Quadrature encoder mode 1, x2 mode- Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

00010:Quadrature encoder mode 2, x2 mode - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

00011:Quadrature encoder mode 3, x4 mode - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

00100:Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

00101:Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

00110:Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

00111:External Clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

01000:Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

01001:Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

01010:Encoder mode: Clock plus direction, x2 mode.

01011:Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P

01100:Encoder mode: Directional Clock, x2 mode.

01101:Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

01110:Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

01111:Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

10000:Quadrature encoder with built-in debouncer: x2 mode.

10001:Quadrature encoder with built-in debouncer: x4 mode.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

34.6.4 TIM DMA/interrupt enable register (TIM_DIER)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction Change interrupt disabled
1: Direction Change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index Change interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bit 12 **CC4DE**: Capture/compare 4 DMA request enable
0: CC4 DMA request disabled
1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bit 4 **CC4IE**: Capture/compare 4 interrupt enable
0: CC4 interrupt disabled
1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

34.6.5 TIM status register (TIM_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TI4FS	TI3FS	TI2FS	TI1FS	Res.	Res.	ODS	UIOVRF	TERRF	IERRF	DIRF	IDXF	Res.	CC7IF	CC6IF	CC5IF
r	r	r	r			r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0		rc_w0	rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B2GF	BGF	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bit 31 **TI4FS**: tim_ti4f status

This flag indicates the value of the tim_ti4f signal (after the digital filtering stage), for polling purpose.
0: tim_ti4f signal is low
1: tim_ti4f signal is high

Bit 30 **TI3FS**: tim_ti3f status

This flag indicates the value of the tim_ti3f signal (after the digital filtering stage), for polling purpose.
0: tim_ti3f signal is low
1: tim_ti3f signal is high

Bit 29 **TI2FS**: tim_ti2f status

This flag indicates the value of the tim_ti2f signal (after the digital filtering stage), for polling purpose.
0: tim_ti2f signal is low
1: tim_ti2f signal is high

Bit 28 **TI1FS**: tim_ti1f status

This flag indicates the value of the tim_ti1f signal (after the digital filtering stage), for polling purpose.
0: tim_ti1f signal is low
1: tim_ti1f signal is high

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **ODS**: Output disable status

This bit indicates the global output state when a break or break2 event occured, to discriminate the source.
0: Idle state: break was triggered (or MOE was written to 0), and OIS states are forced on tim_ocx and tim_ocxn outputs
1: Disabled state: break2 was triggered, tim_ocx and tim_ocxn outputs are inactive

Bit 24 **UIOVRF**: Update Interrupt Overrun flag

This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.
0: No Update Interrupt Overrun
1: Update Interrupt Overrun

Bit 23 **TERRF**: Transition error interrupt flag

This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag

This flag is set by hardware when an index error is detected. It is cleared by software by writing it to 0.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag

This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIM_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to 0.
0: No index event occurred.
1: An index event has occurred

Bit 19 Reserved, must be kept at reset value.

Bit 18 **CC7IF**: Compare 7 interrupt flag
Refer to CC1IF description
Note: Channel 7 can only be configured as output.

Bit 17 **CC6IF**: Compare 6 interrupt flag
Refer to CC1IF description
Note: Channel 6 can only be configured as output.

Bit 16 **CC5IF**: Compare 5 interrupt flag
Refer to CC1IF description
Note: Channel 5 can only be configured as output.

Bit 15 **B2GF**: Break 2 generation flag
This flag is set by hardware as soon as a break 2 event is forced by software, using the B2G bit. It is useful to distinguish hardware-triggered from software-triggered break events. It is cleared by software by writing it to 0.
0: No break 2 event occurred.
1: A break 2 event was forced with the B2G bit

Bit 14 **BGF**: Break generation flag
This flag is set by hardware as soon as a break event is forced by software, using the BG bit. It is useful to distinguish hardware-triggered from software-triggered break events. It is cleared by software by writing it to 0.
0: No break event occurred.
1: A break event was forced with the BG bit

Bit 13 **SBIF**: System break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE = 1 in the TIM_DIER register.

Bit 12 **CC4OF**: Capture/compare 4 overcapture flag
Refer to CC1OF description

Bit 11 **CC3OF**: Capture/compare 3 overcapture flag
Refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIM_CCR1 register while CC1IF flag was already set

Bit 8 **B2IF**: Break 2 interrupt flag

This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.

0: No break event occurred.

1: An active level has been detected on the break 2 input. An interrupt is generated if BIE = 1 in the TIM_DIER register.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input. An interrupt is generated if BIE = 1 in the TIM_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIM_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIM_CNT matches the content of the TIM_CCR1 register. When the content of TIM_CCR1 is greater than the content of TIM_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIM_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIM_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIM_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
–At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIM_CR1 register.
–When CNT is reinitialized by software using the UG bit in TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.
–When CNT is reinitialized by a trigger event (refer to *Section 34.6.3: TIM slave mode control register (TIM_SMCR)*), if URS = 0 and UDIS = 0 in the TIM_CR1 register.

34.6.6 TIM event generation register (TIM_EGR)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							w	w	w	w	w	w	w	w	w

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **B2G**: Break 2 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIM_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/compare control update generation
This bit can be set by software, it is automatically cleared by hardware
0: No action
1: CCxE, CCxNE and OCxM bits update (providing CCPC bit is set)
Note: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIM_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (upcounting), else it takes the autoreload value (TIM_ARR) if DIR = 1 (downcounting).

34.6.7 TIM capture/compare mode register 1 (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode:

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter
Refer to IC1F[3:0] bit description

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler
Refer to IC1PSC[1:0] bit description

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2
10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS2}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS2}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS2}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS2}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS2}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS2}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS2}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS2}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS2}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS2}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS2}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS2}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS2}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIM_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 Selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

34.6.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC2M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]	
						rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

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

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC2CE:** Output compare 2 clear enable

0: tim_oc2ref is not affected by the tim_ocref_clr_int signal

1: tim_oc2ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 34.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 25:24, 14:12 **OC2M[4:0]**: Output compare 2 mode

Refer to OC1M[4:0] bit description

Bit 11 **OC2PE**: Output compare 2 preload enable

0: Preload register on TIM_CCR2 disabled. TIM_CCR2 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR2 enabled. Read/Write operations access the preload register. TIM_CCR2 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 10 **OC2FE**: Output compare 2 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC2 behaves normally depending on counter and CCR2 values even when the trigger is ON. The minimum delay to activate CC2 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC2 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC2 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bit 7 **OC1CE:** Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int signal

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 34.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC1M[4:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

00000:Frozen - The comparison between the output compare register TIM_CCR1 and the counter TIM_CNT has no effect on the outputs.This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

00001:Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00010:Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00011:Toggle - tim_oc1ref toggles when TIM_CNT = TIM_CCR1.

00100:Force inactive level - tim_oc1ref is forced low.

00101:Force active level - tim_oc1ref is forced high.

00110:PWM mode 1 - In upcounting, channel 1 is active as long as TIM_CNT<TIM_CCR1 else inactive. In downcounting, channel 1 is inactive (tim_oc1ref = 0) as long as TIM_CNT>TIM_CCR1 else active (tim_oc1ref = 1).

00111:PWM mode 2 - In upcounting, channel 1 is inactive as long as TIM_CNT<TIM_CCR1 else active. In downcounting, channel 1 is active as long as TIM_CNT>TIM_CCR1 else inactive.

01000:Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001:Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01100-01101:Combined PWM mode 1 and 2. Refer to *Table 382* for details.

01110-01111:Asymmetric PWM mode 1 and 2. Refer to *Table 381* for details.

10000-10001:Combined PWM mode 3 and 4. Refer to *Table 382* for details.

10010-11001: Asymmetric PWM mode 3 to 10. Refer to *Table 381* for details.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Note: On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIM_CCR1 disabled. TIM_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR1 enabled. Read/Write operations access the preload register. TIM_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

34.6.9 TIM capture/compare mode register 2 (TIM_CCMR2)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter
Refer to IC3F[3:0] bit description

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler
Refer to IC3PSC[1:0] bit description

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIM_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter
This bitfield defines the frequency used to sample tim_ti3 input and the length of the digital filter applied to tim_ti3. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:
0000: No filter, sampling is done at f_{DTS2}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS2}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS2}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS2}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS2}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS2}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS2}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS2}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS2}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS2}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS2}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS2}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS2}/32$, N = 8

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler
This bitfield defines the ratio of the prescaler acting on CC3 input (tim_ic3). The prescaler is reset as soon as CC3E = 0 (TIM_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIM_CCER).

34.6.10 TIM capture/compare mode register 2 [alternate] (TIM_CCMR2)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC4M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC3M[4:3]	
						rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

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

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

0: tim_oc4ref is not affected by the tim_ocref_clr_int signal

1: tim_oc4ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 34.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 25:24, 14:12 **OC4M[4:0]**: Output compare 4 mode

Refer to OC3M[4:0] bit description

Bit 11 **OC4PE**: Output compare 4 preload enable

0: Preload register on TIM_CCR4 disabled. TIM_CCR4 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR4 enabled. Read/Write operations access the preload register. TIM_CCR4 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 10 **OC4FE**: Output compare 4 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC4 behaves normally depending on counter and CCR4 values even when the trigger is ON. The minimum delay to activate CC4 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC4 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC4 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4

10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3

11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIM_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable

0: tim_oc3ref is not affected by the tim_ocref_clr_int signal

1: tim_oc3ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 34.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC3M[4:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

00000:Frozen - The comparison between the output compare register TIM_CCR3 and the counter TIM_CNT has no effect on the outputs.(this mode is used to generate a timing base).

00001:Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIM_CNT matches the capture/compare register 3 (TIM_CCR3).

00010:Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIM_CNT matches the capture/compare register 3 (TIM_CCR3).

00011:Toggle - tim_oc3ref toggles when TIM_CNT = TIM_CCR3.

00100:Force inactive level - tim_oc3ref is forced low.

00101:Force active level - tim_oc3ref is forced high.

00110:PWM mode 1 - In upcounting, channel 3 is active as long as TIM_CNT<TIM_CCR3 else inactive. In downcounting, channel 3 is inactive (tim_oc3ref = 0) as long as TIM_CNT>TIM_CCR3 else active (tim_oc3ref = 1).

00111:PWM mode 2 - In upcounting, channel 3 is inactive as long as TIM_CNT<TIM_CCR3 else active. In downcounting, channel 3 is active as long as TIM_CNT>TIM_CCR3 else inactive.

01000:Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001:Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01010:Pulse on compare: a pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIM_ECR.

01011:Direction output. The tim_oc3ref signal is overridden by a copy of the DIR bit.

01100-01101:Combined PWM mode 1 and 2. Refer to *Table 382* for details.

01110-01111:Asymmetric PWM mode 1 and 2. Refer to *Table 381* for details.

10000-10001:Combined PWM mode 3 and 4. Refer to *Table 382* for details.

10010-11001: Asymmetric PWM mode 3 to 10. Refer to *Table 381* for details.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Note: On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC3PE**: Output compare 3 preload enable

0: Preload register on TIM_CCR3 disabled. TIM_CCR3 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR3 enabled. Read/Write operations access the preload register. TIM_CCR3 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC3FE**: Output compare 3 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC3 behaves normally depending on counter and CCR3 values even when the trigger is ON. The minimum delay to activate CC3 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC3 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC3 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3

10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4

11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIM_CCER).

34.6.11 TIM capture/compare enable register (TIM_CCER)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CC7P	CC7E	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
						rw	rw			rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 25 **CC7P**: Capture/compare 7 output polarity
Refer to CC1P description

Bit 24 **CC7E**: Capture/compare 7 output enable
Refer to CC1E description

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **CC6P**: Capture/compare 6 output polarity
Refer to CC1P description

Bit 20 **CC6E**: Capture/compare 6 output enable
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/compare 5 output polarity
Refer to CC1P description

Bit 16 **CC5E**: Capture/compare 5 output enable
Refer to CC1E description

Bit 15 **CC4NP**: Capture/compare 4 complementary output polarity
Refer to CC1NP description

Bit 14 **CC4NE**: Capture/compare 4 complementary output enable
Refer to CC1NE description

Bit 13 **CC4P**: Capture/compare 4 output polarity
Refer to CC1P description

Bit 12 **CC4E**: Capture/compare 4 output enable
Refer to CC1E description

Bit 11 **CC3NP**: Capture/compare 3 complementary output polarity
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/compare 3 complementary output enable
Refer to CC1NE description

Bit 9 **CC3P**: Capture/compare 3 output polarity
Refer to CC1P description

Bit 8 **CC3E**: Capture/compare 3 output enable
Refer to CC1E description

Bit 7 **CC2NP**: Capture/compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/compare 2 complementary output enable
Refer to CC1NE description

Bit 5 **CC2P**: Capture/compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity
CC1 channel configured as output:
0: tim_oc1n active high.
1: tim_oc1n active low.
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1.
Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (channel configured as output).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Note: In input mode, this bit is preloaded as per CCPC bit in TIM_CR2.

Bit 2 **CC1NE**: Capture/compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 1 **CC1P**: Capture/compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0:non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1:inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0:the configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Note: In input mode, this bit is preloaded as per CCPC bit in TIM_CR2.

Bit 0 **CC1E**: Capture/compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 390* for details.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Table 390. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature

<table>
<tr><th colspan="5">Control bits</th><th colspan="2">Output states(1)</th></tr>
<tr><th>MOE bit</th><th>OSSI bit</th><th>OSSR bit</th><th>CCxE bit</th><th>CCxNE bit</th><th>tim_ocx output state</th><th>tim_ocxn output state</th></tr>
<tr><td rowspan="8">1</td><td rowspan="8">X</td><td>0</td><td>0</td><td>0</td><td colspan="2">Output disabled (not driven by the timer: Hi-Z)
tim_ocx = 0, tim_ocxn = 0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td colspan="2">Off-State (output enabled with inactive state)
tim_ocx=CCxP, tim_ocxn=CCxNP</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Output disabled (not driven by the timer: Hi-Z)
tim_ocx = 0</td><td>tim_ocxref + Polarity tim_ocxn = tim_ocxref xor CCxNP</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>tim_ocxref + Polarity
tim_ocx = tim_ocxref xor CCxP</td><td>Output Disabled (not driven by the timer: Hi-Z)
tim_ocxn = 0</td></tr>
<tr><td>X</td><td>1</td><td>1</td><td>OCREF + Polarity + dead-time</td><td>Complementary to OCREF (not OCREF) + Polarity + dead-time</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Off-State (output enabled with inactive state)
tim_ocx = CCxP</td><td>tim_ocxref + Polarity
tim_ocxn = tim_ocxref x or CCxNP</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>tim_ocxref + Polarity
tim_ocx = tim_ocxref xor CCxP</td><td>Off-State (output enabled with inactive state)
tim_ocxn = CCxNP</td></tr>
<tr><td rowspan="6">0</td><td>0</td><td rowspan="6">X</td><td>X</td><td>X</td><td colspan="2" rowspan="2">Output disabled (not driven by the timer: Hi-Z).</td></tr>
<tr><td rowspan="5">1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td colspan="2" rowspan="3">Off-State (output enabled with inactive state)
Asynchronously: tim_ocx = CCxP, tim_ocxn = CCxNP (if tim_brk or tim_brk2 is triggered).
Then (this is valid only if tim_brk is triggered), if the clock is present: tim_ocx = OISx and tim_ocxn = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and tim_ocxn both in active state (may cause a short circuit when driving switches in half-bridge configuration).
Note: tim_brk2 can only be used if OSSI = OSSR = 1.</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and the GPIO registers.

34.6.12 TIM counter (TIM_CNT)

Address offset: 0x024

Reset value: 0x0000 0000

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

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

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIM_ISR register. If the UIFREMAP bit in the TIM_CR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

34.6.13 TIM prescaler (TIM_PSC)

Address offset: 0x028

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency ($f_{tim_cnt_ck}$) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM_EGR register or through trigger controller when configured in “reset mode”).

34.6.14 TIM autoreload register (TIM_ARR)

Address offset: 0x02C

Reset value: 0x0000 FFFF

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 34.3.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the autoreload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the autoreload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

34.6.15 TIM repetition counter register (TIM_RCR)

Address offset: 0x030

Reset value: 0x0000

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

Bits 15:0 **REP[15:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

– the number of PWM periods in edge-aligned mode

– the number of half PWM period in center-aligned mode.

34.6.16 TIM capture/compare register 1 (TIM_CCR1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIM_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

34.6.17 TIM capture/compare register 2 (TIM_CCR2)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input: CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIM_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

34.6.18 TIM capture/compare register 3 (TIM_CCR3)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR3[19:0]**: Capture/compare value

If channel CC3 is configured as output: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIM_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc3 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR3[19:4]. The CCR3[3:0] bitfield contains the dithered part.
If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIM_CCR3 register is read-only and cannot be programmed.
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR3[19:4]. The CCR3[3:0] bits are reset.

34.6.19 TIM capture/compare register 4 (TIM_CCR4)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR4[19:0]**: Capture/compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIM_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIM_CNT and signalled on tim_oc4 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR4[19:4]. The CCR4[3:0] bitfield contains the dithered part.
If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIM_CCR4 register is read-only and cannot be programmed.
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR4[15:0]. The CCR4[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR4[19:4]. The CCR4[3:0] bits are reset.

34.6.20 TIM break and dead-time register (TIM_BDTR)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits BKBID/BK2BID/BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIM_BDTR register.

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **BK2BID**: Break2 bidirectional
Refer to BKBID description

Bit 28 **BKBID**: Break bidirectional
0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 **BK2DSRM**: Break2 disarm
Refer to BKDSRM description

Bit 26 **BKDSRM**: Break disarm
0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 25 **BK2P**: Break 2 polarity
0: Break input tim_brk2 is active low
1: Break input tim_brk2 is active high
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BK2E bit is set.
Note: Any write operation to this bit takes a delay of 1 tim_ker_ck clock cycle to become effective.

Bit 24 **BK2E**: Break 2 enable
This bit enables the complete break 2 protection, see *Figure 565: Break and Break2 circuitry overview*).
0: Break2 function disabled
1: Break2 function enabled
Note: The BRKIN2 must only be used with OSSR = OSSI = 1.
Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).
Note: Any write operation to this bit takes a delay of 1 tim_ker_ck clock cycle to become effective.

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bitfield defines the frequency used to sample tim_brk2 input and the length of the digital filter applied to tim_brk2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk2 acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BK2E bit is set.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (tim_brk or tim_brk2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled
In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIM_CCER register).
See OC/OCN enable description for more details (*Section 34.6.11: TIM capture/compare enable register (TIM_CCER)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs tim_brk and tim_brk2 is active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Note: Any write operation to this bit takes a delay of 1 tim_ker_ck clock cycle to become effective.

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to tim_sys_brk and BKIN sources, as per *Figure 565: Break and Break2 circuitry overview*).

0: Break function disabled

1: Break function enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Note: Any write operation to this bit takes a delay of 1 tim_ker_ck clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 34.6.11: TIM capture/compare enable register (TIM_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **OSSI**: Off-state selection for idle mode

This bit is used when MOE = 0 due to a break event or by a software write, on channels configured as outputs.
See OC/OCN enable description for more details (*Section 34.6.11: TIM capture/compare enable register (TIM_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIM_BDTR register, OISx and OISxN bits in TIM_CR2 register and BKBID/BK2BID/BKE/BKP/AOE bits in TIM_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIM_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIM_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIM_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5] = 0xx => DT = DTG[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$.
DTG[7:5] = 10x => DT = (64+DTG[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$.
DTG[7:5] = 110 => DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$.
DTG[7:5] = 111 => DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$.
Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

34.6.21 TIM capture/compare register 5 (TIM_CCR5)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	GC5C4	GC5C4O	GC5C3O	GC5C2O	GC5C1O	Res.	Res.	Res.	Res.	CCR5[19:16]			
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group channel 5 and channel 3

Distortion on channel 3 output:

0: No effect of tim_oc5ref on tim_oc3refc

1: tim_oc3refc is the logical AND of tim_oc3ref and tim_oc5ref

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 30 **GC5C2**: Group channel 5 and channel 2

Distortion on channel 2 output:

0: No effect of tim_oc5ref on tim_oc2refc

1: tim_oc2refc is the logical AND of tim_oc2ref and tim_oc5ref

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 29 **GC5C1**: Group channel 5 and channel 1

Distortion on channel 1 output:

0: No effect of oc5ref on oc1refc

1: oc1refc is the logical AND of oc1ref and oc5ref

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 28 **GC5C4**: Group channel 5 and channel 4

Distortion on channel 1 output:

0: No effect of oc5ref on oc4refc

1: oc4refc is the logical AND of oc4ref and oc5ref

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 27 **GC5C4O**: Group channel 5 and channel 4 (ORed)

Distortion on channel 4 output:

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

0: No effect of oc5ref on oc4refc

1: oc4refc is the logical OR of oc4ref and oc5ref

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 26 **GC5C3O**: Group channel 5 and channel 3 (ORed)

Distortion on channel 3 output:

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

0: No effect of oc5ref on oc3refc

1: oc4refc is the logical OR of oc3ref and oc5ref

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 25 **GC5C2O**: Group channel 5 and channel 2 (ORed)
Distortion on channel 2 output:
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).
0: No effect of oc5ref on oc2refc
1: oc4refc is the logical OR of oc2ref and oc5ref
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 24 **GC5C1O**: Group channel 5 and channel 1 (ORed)
Distortion on channel 1 output:
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).
0: No effect of oc5ref on oc1refc
1: oc4refc is the logical OR of oc1ref and oc5ref
Note: it is also possible to apply this distortion on combined PWM signals.

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

Bits 19:0 **CCR5[19:0]**: compare 5 value
CCR5 is the value to be loaded in the actual compare 5 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIM_CCMR3 register (bit OC5PE). Else the preload value is copied in the active compare 5 register when an update event occurs.
The active compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc5 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR5[15:0]. The CCR5[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR5[19:4]. The CCR5[3:0] bitfield contains the dithered part.

34.6.22 TIM capture/compare register 6 (TIM_CCR6)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR6[19:0]**: Compare 6 value

CCR6 is the value to be loaded in the actual compare 6 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR3 register (bit OC6PE). Else the preload value is copied in the active compare 6 register when an update event occurs.

The active compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc6 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR6[15:0]. The CCR6[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR6[19:4]. The CCR6[3:0] bitfield contains the dithered part.

34.6.23 TIM capture/compare mode register 3 (TIM_CCMR3)

Address offset: 0x050

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channels 5 and 6 can only be configured in output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC6M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC5M[4:3]	
						rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6CE	OC6M[2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

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

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable

0: tim_oc6ref is not affected by the tim_ocref_clr_int signal

1: tim_oc6ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 34.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 25:24, 14:12 **OC6M[4:0]**: Output compare 6 mode

Refer to OC5M[4:0] bit description

Bit 11 **OC6PE**: Output compare 6 preload enable

0: Preload register on TIM_CCR6 disabled. TIM_CCR6 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR6 enabled. Read/Write operations access the preload register. TIM_CCR6 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 10 **OC6FE**: Output compare 6 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC6 behaves normally depending on counter and CCR6 values even when the trigger is ON. The minimum delay to activate CC6 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC6 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC6 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **OC5CE:** Output compare 5 clear enable

0: tim_oc5ref is not affected by the tim_ocref_clr_int signal

1: tim_oc5ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 34.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC5M[4:0]**: Output compare 5 mode

These bits define the behavior of the output reference signal tim_oc5ref5. tim_oc5ref is active high whereas tim_oc5 active level depends on CC5P bit.

00000:Frozen - The comparison between the output compare register TIM_CCR5 and the counter TIM_CNT has no effect on the outputs.(this mode is used to generate a timing base).

00001:Set channel 5 to active level on match. tim_oc5ref signal is forced high when the counter TIM_CNT matches the capture/compare register 5 (TIM_CCR5).

00010:Set channel 5 to inactive level on match. tim_oc5ref signal is forced low when the counter TIM_CNT matches the capture/compare register 5 (TIM_CCR5).

00011:Toggle - tim_oc5ref toggles when TIM_CNT = TIM_CCR5.

00100:Force inactive level - tim_oc5ref is forced low.

00101:Force active level - tim_oc5ref is forced high.

00110:PWM mode 1 - In upcounting, channel 5 is active as long as TIM_CNT<TIM_CCR5 else inactive. In downcounting, channel 5 is inactive (tim_oc5ref = 0) as long as TIM_CNT>TIM_CCR5 else active (tim_oc5ref = 1).

00111:PWM mode 2 - In upcounting, channel 5 is inactive as long as TIM_CNT<TIM_CCR5 else active. In downcounting, channel 5 is active as long as TIM_CNT>TIM_CCR5 else inactive.

01000:Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001:Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01010-01011:Reserved.

01100-01101:Combined PWM mode 1 and 2. Refer to *Table 382* for details.

01110-01111:Asymmetric PWM mode 1 and 2. Refer to *Table 381* for details.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Bit 3 **OC5PE**: Output compare 5 preload enable

0: Preload register on TIM_CCR5 disabled. TIM_CCR5 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR5 enabled. Read/Write operations access the preload register. TIM_CCR5 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC5FE**: Output compare 5 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC5 behaves normally depending on counter and CCR5 values even when the trigger is ON. The minimum delay to activate CC5 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC5 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC5 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

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

34.6.24 TIM timer deadtime register 2 (TIM_DTR2)

Address offset: 0x054

Reset value: 0x0000 0000

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

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

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx => DTF = DTGF[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$.

DTGF[7:5] = 10x => DTF = (64+DTGF[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$.

DTGF[7:5] = 110 => DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$.

DTGF[7:5] = 111 => DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$.

Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

34.6.25 TIM timer encoder control register (TIM_ECR)

Address offset: 0x058

Reset value: 0x0000 0000

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

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

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[4:0] = 00001, 00010, 00011, 01110, 01111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[4:0] = 01010, 01011, 01100, 01101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 **IBLK[1:0]**: Index blanking

This bit indicates if the Index event is conditioned by the tim_ti3 or tim_ti4 input

00: Index always active
01: Index disabled when tim_ti3 input is active, as per CC3P bitfield
10: Index disabled when tim_ti4 input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction

This bit indicates in which direction the Index event resets the counter.

00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

34.6.26 TIM timer input selection register (TIM_TISEL)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIM_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 34.3.2: TIM1/TIM8/TIM20 pins and internal signals* for interconnects list.

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

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIM_CH2
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 34.3.2: TIM1/TIM8/TIM20 pins and internal signals* for interconnects list.

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

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIM_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 34.3.2: TIM1/TIM8/TIM20 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIM_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 34.3.2: TIM1/TIM8/TIM20 pins and internal signals* for interconnects list.

34.6.27 TIM alternate function option register 1 (TIM_AF1)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKCM P20E	BKCM P19E	BKCM P18E	BKCM P17E	BKCM P16E	BKCM P15E	BKCM P14E	BKCM P13E	BKCM P12E	BKCM P11E	BKCM P10E	BKCM P9E	Res.	ETRSEL[4:2]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		BKCM P4P	BKCM P3P	BKCM P2P	BKCM P1P	BKINP	BKCM P8E	BKCM P7E	BKCM P6E	BKCM P5E	BKCM P4E	BKCM P3E	BKCM P2E	BKCM P1E	BKINE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BKCMP20E**: tim_brk_cmp20 enable

This bit enables the tim_brk_cmp20 for the timer's tim_brk input. tim_brk_cmp20 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp20 input disabled

1: tim_brk_cmp20 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 30 **BKCMP19E**: tim_brk_cmp19 enable

This bit enables the tim_brk_cmp19 for the timer's tim_brk input. tim_brk_cmp19 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp19 input disabled

1: tim_brk_cmp19 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 29 **BKCMP18E**: tim_brk_cmp18 enable

This bit enables the tim_brk_cmp18 for the timer's tim_brk input. tim_brk_cmp18 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp18 input disabled

1: tim_brk_cmp18 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 28 **BKCMP17E**: tim_brk_cmp17 enable

This bit enables the tim_brk_cmp17 for the timer's tim_brk input. tim_brk_cmp17 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp17 input disabled

1: tim_brk_cmp17 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 27 **BKCMP16E**: tim_brk_cmp16 enable

This bit enables the tim_brk_cmp16 for the timer's tim_brk input. tim_brk_cmp16 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp16 input disabled

1: tim_brk_cmp16 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 26 **BKCMP15E**: tim_brk_cmp15 enable

This bit enables the tim_brk_cmp15 for the timer's tim_brk input. tim_brk_cmp15 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp15 input disabled

1: tim_brk_cmp15 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 25 **BKCMP14E**: tim_brk_cmp14 enable

This bit enables the tim_brk_cmp14 for the timer's tim_brk input. tim_brk_cmp14 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp14 input disabled

1: tim_brk_cmp14 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 24 **BKCMP13E**: tim_brk_cmp13 enable

This bit enables the tim_brk_cmp13 for the timer's tim_brk input. tim_brk_cmp13 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp13 input disabled

1: tim_brk_cmp13 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 23 **BKCMP12E**: tim_brk_cmp12 enable

This bit enables the tim_brk_cmp12 for the timer's tim_brk input. tim_brk_cmp12 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp12 input disabled

1: tim_brk_cmp12 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 22 **BKCMP11E**: tim_brk_cmp11 enable

This bit enables the tim_brk_cmp11 for the timer's tim_brk input. tim_brk_cmp11 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp11 input disabled

1: tim_brk_cmp11 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 21 **BKCMP10E**: tim_brk_cmp10 enable

This bit enables the tim_brk_cmp10 for the timer's tim_brk input. tim_brk_cmp10 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp10 input disabled

1: tim_brk_cmp10 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 20 **BKCMP9E**: tim_brk_cmp9 enable

This bit enables the tim_brk_cmp9 for the timer's tim_brk input. tim_brk_cmp9 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp9 input disabled

1: tim_brk_cmp9 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 19 Reserved, must be kept at reset value.

Bits 18:14 **ETRSEL[4:0]**: etr_in source selection

These bits select the etr_in input source.

00000:tim_etr0: TIM_ETR input

00001:tim_etr1

...

11111:tim_etr31

Refer to *Section 34.3.2: TIM1/TIM8/TIM20 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp4 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp3 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp2 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp1 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 9 **BKINP**: TIM_BKIN input polarity

This bit selects the TIM_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIM_BKIN input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: TIM_BKIN input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **BKINE**: TIM_BKIN input enable

This bit enables the TIM_BKIN alternate function input for the timer's tim_brk input. TIM_BKIN input is 'ORed' with the other tim_brk sources.

0: TIM_BKIN input disabled

1: TIM_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Note: Refer to Section 34.3.2: TIM1/TIM8/TIM20 pins and internal signals for product specific implementation.

34.6.28 TIM alternate function register 2 (TIM_AF2)

Address offset: 0x064

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BK2CMP20E	BK2CMP19E	BK2CMP18E	BK2CMP17E	BK2CMP16E	BK2CMP15E	BK2CMP14E	BK2CMP13E	BK2CMP12E	BK2CMP11E	BK2CMP10E	BK2CMP9E	OCRSEL[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BK2CMP20E**: tim_brk2_cmp20 enable

This bit enables the tim_brk2_cmp20 for the timer's tim_brk2 input. tim_brk2_cmp20 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp20 input disabled

1: tim_brk2_cmp20 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 30 **BK2CMP19E**: tim_brk2_cmp19 enable

This bit enables the tim_brk2_cmp19 for the timer's tim_brk2 input. tim_brk2_cmp19 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp19 input disabled

1: tim_brk2_cmp19 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 29 **BK2CMP18E**: tim_brk2_cmp18 enable

This bit enables the tim_brk2_cmp18 for the timer's tim_brk2 input. tim_brk2_cmp18 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp18 input disabled

1: tim_brk2_cmp18 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 28 **BK2CMP17E**: tim_brk2_cmp17 enable

This bit enables the tim_brk2_cmp17 for the timer's tim_brk2 input. tim_brk2_cmp17 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp17 input disabled

1: tim_brk2_cmp17 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 27 **BK2CMP16E**: tim_brk2_cmp16 enable

This bit enables the tim_brk2_cmp16 for the timer's tim_brk2 input. tim_brk2_cmp16 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp16 input disabled

1: tim_brk2_cmp16 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 26 **BK2CMP15E**: tim_brk2_cmp15 enable

This bit enables the tim_brk2_cmp15 for the timer's tim_brk2 input. tim_brk2_cmp15 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp15 input disabled

1: tim_brk2_cmp15 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 25 **BK2CMP14E**: tim_brk2_cmp14 enable

This bit enables the tim_brk2_cmp14 for the timer's tim_brk2 input. tim_brk2_cmp14 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp14 input disabled

1: tim_brk2_cmp14 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 24 **BK2CMP13E**: tim_brk2_cmp13 enable

This bit enables the tim_brk2_cmp13 for the timer's tim_brk2 input. tim_brk2_cmp13 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp13 input disabled

1: tim_brk2_cmp13 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 23 **BK2CMP12E**: tim_brk2_cmp12 enable

This bit enables the tim_brk2_cmp12 for the timer's tim_brk2 input. tim_brk2_cmp12 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp12 input disabled

1: tim_brk2_cmp12 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 22 **BK2CMP11E**: tim_brk2_cmp11 enable

This bit enables the tim_brk2_cmp11 for the timer's tim_brk2 input. tim_brk2_cmp11 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp11 input disabled

1: tim_brk2_cmp11 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 21 **BK2CMP10E**: tim_brk2_cmp10 enable

This bit enables the tim_brk2_cmp10 for the timer's tim_brk2 input. tim_brk2_cmp10 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp10 input disabled

1: tim_brk2_cmp10 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 20 **BK2CMP9E**: tim_brk2_cmp9 enable

This bit enables the tim_brk2_cmp9 for the timer's tim_brk2 input. tim_brk2_cmp9 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp9 input disabled

1: tim_brk2_cmp9 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 19:16 **OCRSEL[3:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

0000:tim_ocref_clr0

0001:tim_ocref_clr1

...

1111: tim_ocref_clr15

Refer to *Section 34.3.2: TIM1/TIM8/TIM20 pins and internal signals* for product specific information.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

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

Bit 13 **BK2CMP4P**: tim_brk2_cmp4 input polarity

This bit selects the tim_brk2_cmp4 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp4 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp4 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 12 **BK2CMP3P**: tim_brk2_cmp3 input polarity

This bit selects the tim_brk2_cmp3 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp3 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp3 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 11 **BK2CMP2P**: tim_brk2_cmp2 input polarity

This bit selects the tim_brk2_cmp2 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **BK2CMP1P**: tim_brk2_cmp1 input polarity

This bit selects the tim_brk2_cmp1 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp1 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp1 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 9 **BK2INP**: TIM_BKIN2 input polarity

This bit selects the TIM_BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: TIM_BKIN2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: TIM_BKIN2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 8 **BK2CMP8E**: tim_brk2_cmp8 enable

This bit enables the tim_brk2_cmp8 for the timer's tim_brk2 input. tim_brk2_cmp8 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp8 input disabled

1: tim_brk2_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 7 **BK2CMP7E**: tim_brk2_cmp7 enable

This bit enables the tim_brk2_cmp7 for the timer's tim_brk2 input. tim_brk2_cmp7 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp7 input disabled

1: tim_brk2_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 6 **BK2CMP6E**: tim_brk2_cmp6 enable

This bit enables the tim_brk2_cmp6 for the timer's tim_brk2 input. tim_brk2_cmp6 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp6 input disabled

1: tim_brk2_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 5 **BK2CMP5E**: tim_brk2_cmp5 enable

This bit enables the tim_brk2_cmp5 for the timer's tim_brk2 input. tim_brk2_cmp5 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp5 input disabled

1: tim_brk2_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 4 **BK2CMP4E**: tim_brk2_cmp4 enable

This bit enables the tim_brk2_cmp4 for the timer's tim_brk2 input. tim_brk2_cmp4 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp4 input disabled

1: tim_brk2_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 3 **BK2CMP3E**: tim_brk2_cmp3 enable

This bit enables the tim_brk2_cmp3 for the timer's tim_brk2 input. tim_brk2_cmp3 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp3 input disabled

1: tim_brk2_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 2 **BK2CMP2E**: tim_brk2_cmp2 enable

This bit enables the tim_brk2_cmp2 for the timer's tim_brk2 input. tim_brk2_cmp2 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp2 input disabled

1: tim_brk2_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **BK2CMP1E**: tim_brk2_cmp1 enable

This bit enables the tim_brk2_cmp1 for the timer's tim_brk2 input. tim_brk2_cmp1 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp1 input disabled

1: tim_brk2_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **BK2INE**: TIM_BKIN2 input enable

This bit enables the TIM_BKIN2 alternate function input for the timer's tim_brk2 input. TIM_BKIN2 input is 'ORed' with the other tim_brk2 sources.

0: TIM_BKIN2 input disabled

1: TIM_BKIN2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Note: Refer to Section 34.3.2: TIM1/TIM8/TIM20 pins and internal signals for product specific implementation.

34.6.29 TIM capture/compare register 7 (TIM_CCR7)

Address offset: 0x070

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR7[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR7[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR7[19:0]**: Capture/compare 7 value

CCR7 is the value to be loaded in the actual compare 7 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR4 register (bit OC7PE). Else the preload value is copied in the active compare 7 register when an update event occurs.

The active compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc7 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR7[15:0]. The CCR7[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR7[19:4]. The CCR7[3:0] bitfield contains the dithered part.

34.6.30 TIM capture/compare mode register 4 (TIM_CCMR4)

Address offset: 0x078

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channel 7 can only be configured in output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC7M[4:3]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC7CE	OC7M[2:0]			OC7PE	OC7FE	Res.	Res.
								rw	rw	rw	rw	rw	rw		

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

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OC7CE:** Output compare 7 clear enable

0: tim_oc7ref is not affected by the tim_ocref_clr_int signal

1: tim_oc7ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 34.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC7M[4:0]**: Output compare 7 mode

These bits define the behavior of the output reference signal tim_ocref7. tim_oc7ref is active high whereas tim_oc7 active level depends on CC7P bit.

00000:Frozen - The comparison between the output compare register TIM_CCR7 and the counter TIM_CNT has no effect on the outputs.(this mode is used to generate a timing base).

00001:Set channel 7 to active level on match. tim_oc7ref signal is forced high when the counter TIM_CNT matches the capture/compare register 7 (TIM_CCR7).

00010:Set channel 7 to inactive level on match. tim_oc7ref signal is forced low when the counter TIM_CNT matches the capture/compare register 7 (TIM_CCR7).

00011:Toggle - tim_oc7ref toggles when TIM_CNT = TIM_CCR7.

00100:Force inactive level - tim_oc7ref is forced low.

00101:Force active level - tim_oc7ref is forced high.

00110:PWM mode 1 - In upcounting, channel 7 is active as long as TIM_CNT<TIM_CCR7 else inactive. In downcounting, channel 7 is inactive (tim_oc7ref = 0) as long as TIM_CNT>TIM_CCR7 else active (tim_oc5ref = 1).

00111:PWM mode 2 - In upcounting, channel 7 is inactive as long as TIM_CNT<TIM_CCR7 else active. In downcounting, channel 7 is active as long as TIM_CNT>TIM_CCR7 else inactive.

01000:Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001:Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Bit 3 **OC7PE**: Output compare 7 preload enable

0: Preload register on TIM_CCR7 disabled. TIM_CCR7 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR7 enabled. Read/Write operations access the preload register. TIM_CCR7 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC7FE**: Output compare 7 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC7 behaves normally depending on counter and CCR7 values even when the trigger is ON. The minimum delay to activate CC7 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC7 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC7 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

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

34.6.31 TIM multilevel protection register 1 (TIM_MPR1)

Address offset: 0x090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BK4NM[1:0]		BK4M[1:0]		BK3NM[1:0]		BK3M[1:0]		BK2NM[1:0]		BK2M[1:0]		BK1NM[1:0]		BK1M[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **BK4NM[1:0]**: Break mode for tim_oc4n output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 13:12 **BK4M[1:0]**: Break mode for tim_oc4 output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 11:10 **BK3NM[1:0]**: Break mode for tim_oc3n output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 9:8 **BK3M[1:0]**: Break mode for tim_oc3 output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 7:6 **BK2NM[1:0]**: Break mode for tim_oc2n output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 5:4 **BK2M[1:0]**: Break mode for tim_oc2 output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 3:2 **BK1NM[1:0]**: Break mode for tim_oc1n output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 1:0 **BK1M[1:0]**: Break mode for tim_oc1 output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

34.6.32 TIM multilevel protection register 2 (TIM_MPR2)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **DBK2[7:0]**: Delayed 2 Break duration

This bitfield defines the duration of the delay in Delayed 2 Break mode.

It is partially preloaded and can be updated during timer operation (new value taken into account after an update event), at the exception of the bits coding for t_{DTG} selection, which are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBK2[6:0] if DBK2[7] = 0
- DBK2[5:0] if DBK2[7:6] = 10
- DBK2[4:0] if DBK2[7:5] = 110 or 111

DBK2[7:5]=0xx => Delay2=DBK2[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK2[7:5]=10x => Delay2=(64+DBK2[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK2[7:5]=110 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK2[7:5]=111 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: Due to internal resynchronization of the break event on tim_ker_ck, the effective delayed break time is the programmed Delay2 plus up to two additional timer clock cycles.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DBK1[7:0]**: Delayed 1 Break duration

This bitfield defines the duration of the delay in Delayed 1 Break mode.

It is partially preloaded and can be updated during timer operation (new value taken into account after an update event), at the exception of the bits coding for t_{DTG} selection, which are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBK1[6:0] if DBK1[7] = 0
- DBK1[5:0] if DBK1[7:6] = 10
- DBK1[4:0] if DBK1[7:5] = 110 or 111

DBK1[7:5]=0xx => Delay1=DBK1[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK1[7:5]=10x => Delay1=(64+DBK1[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK1[7:5]=110 => Delay1=(32+DBK1[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK1[7:5]=111 => Delay1=(32+DBK1[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: Due to internal resynchronization of the break event on tim_ker_ck, the effective delayed break time is the programmed Delay1 plus up to two additional timer clock cycles.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

34.6.33 TIM output override register (TIM_OOR)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	OOS4N	OOS4	OOS3N	OOS3	OOS2N	OOS2	OOS1N	OOS1
			rw					rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **OOC**: Output override control

This bit forces the idle state of the tim_ocx and tim_ocxn outputs to the values defined with OOSx and OOSxN bits.

0: tim_ocx and tim_ocxn outputs are in the default idle state

1: tim_ocx and tim_ocxn outputs are forced in an idle state defined with OOSx/OOSxN bits

Note: This bit can only be modified when the outputs are in idle state (MOE = 0). It is cleared when the MOE bit is set.

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **OOS4N**: Output override state for output tim_oc4n

This bit defines the state for output tim_oc4n when OOC = 1

0: tim_oc4n = 0

1: tim_oc4n = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 6 **OOS4**: Output override state for output tim_oc4

This bit defines the state for output tim_oc4 when OOC = 1

0: tim_oc4 = 0

1: tim_oc4 = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 5 **OOS3N**: Output override state for output tim_oc3n

This bit defines the state for output tim_oc3n when OOC = 1

0: tim_oc3n = 0

1: tim_oc3n = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 4 **OOS3**: Output override state for output tim_oc3

This bit defines the state for output tim_oc3 when OOC = 1

0: tim_oc3 = 0

1: tim_oc3 = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 3 **OOS2N**: Output override state for output tim_oc2n
This bit defines the state for output tim_oc2n when OOC = 1
0: tim_oc2n = 0
1: tim_oc2n = 1
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 2 **OOS2**: Output override state for output tim_oc2
This bit defines the state for output tim_oc2 when OOC = 1
0: tim_oc2 = 0
1: tim_oc2 = 1
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **OOS1N**: Output override state for output tim_oc1n
This bit defines the state for output tim_oc1n when OOC = 1
0: tim_oc1n = 0
1: tim_oc1n = 1
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **OOS1**: Output override state for output tim_oc1
This bit defines the state for output tim_oc1 when OOC = 1
0: tim_oc1 = 0
1: tim_oc1 = 1
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

34.6.34 TIM DMA control register (TIM_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address).

0000: Disabled
0001: Update
0010: CC1
0011: CC2
0100: CC3
0101: CC4
0110: COM
0111: Trigger
Others: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers.

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11111: 32 transfers

Example: Let us consider the following transfer: DBL = 6 & DBA = TIM2_CR1.

–If DBL = 6 and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation:

(TIM_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIM_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIM_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register.

Example:

00000: TIM_CR1
00001: TIM_CR2
00010: TIM_SMCR
...

34.6.35 TIM DMA address for full transfer (TIM_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIM_CR1 address) + (DBA + DMA index) x 4

where TIM_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM_DCR).

34.6.36 TIM1/TIM8/TIM20 register map

TIM1/TIM8/TIM20 registers are mapped as 16-bit addressable registers as described in the table below:

Table 391. TIM1/TIM8/TIM20 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIM_CR1	Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	TGO2PSC[4:0]					Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value					0	0	0	0				0	0	0	0	0				0	0		0	0	0	0	0	0	0	0	0	0
0x004	TIM_CR2	TI3INV	TI2INV	TI1INV	ADSYNC	XORPS	Res.	MMS[3]	MMS2[4:0]				OIS7	OIS6	Res.	OIS5	OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Res.	CCPC
	Reset value	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x008	TIM_SMCR	Res.	Res.	SETPS[3:0]				SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	SMS[4:3]		ETP	ECE	ETPS [1:0]		ETF[3:0]			MSM	TS[2:0]			OCCS	SMS[2:0]			
	Reset value			0	0	0	0	0	0			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	TIM_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value									0	0	0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	TIM_SR	TI4FS	TI3FS	TI2FS	TI1FS	Res.	Res.	ODS	UIOVRF	TERRF	IERRF	DIRF	IDXF	Res.	CC7IF	CC6IF	CC5IF	B2GF	BGF	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value	0	0	0	0			0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	TIM_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0

Table 391. TIM1/TIM8/TIM20 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	TIM_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2 S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]		OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value							0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	TIM_CCMR2	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC3M[4:3]		OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value							0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR2	Res.	Res.	Res	Res.	Res.	Res.	Res	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	TIM_CCER	Res.	Res.	Res.	Res.	Res.	Res.	CC7P	CC7E	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value							0	0			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	TIM_CNT	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	TIM_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	TIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	TIM_RCR	Res.	Res.	Res	Res.	Res.	Res.	Res	Res	Res.	Res.	Res	Res	Res	Res	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	TIM_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	TIM_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	TIM_CCR3	Res.	Res.	Res	Res.	Res.	Res.	Res	Res	Res.	Res.	Res	Res	CCR3[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	TIM_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	TIM_BDTR	Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 391. TIM1/TIM8/TIM20 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x048	TIM_CCR5	GC5C3	GC5C2	GC5C1	GC5C4	GC5C4O	GC5C3O	GC5C2O	GC5C1O	Res.	Res.	Res.	Res.	CCR5[19:0]																			
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	TIM_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	TIM_CCMR3	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC5M[4:3]		OC6CE	OC6M [2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]			OC5PE	OC5FE	Res.	Res.
	Reset value							0	0							0	0	0	0	0	0	0	0			0	0	0	0	0	0		
0x054	TIM_DTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x058	TIM_ECR	Res.	Res.	Res.	Res.	Res.	PWPR SC[2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	IBLK [1:0]		IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x05C	TIM_TISEL	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	TIM_AF1	BKCMP20E	BKCMP19E	BKCMP18E	BKCMP17E	BKCMP16E	BKCMP15E	BKCMP14E	BKCMP13E	BKCMP12E	BKCMP11E	BKCMP10E	BKCMP9E	Res.	ETRSEL [4:0]						BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x064	TIM_AF2	BK2CMP20E	BK2CMP19E	BK2CMP18E	BK2CMP17E	BK2CMP16E	BK2CMP15E	BK2CMP14E	BK2CMP13E	BK2CMP12E	BK2CMP11E	BK2CMP10E	BK2CMP9E	OCRSEL[3:0]				Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x068.. 0x06C	Reserved	Res.																															
0x070	TIM_CCR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR719:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	Reserved	Res.																															
0x078	TIM_CCMR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC7M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC7CE	OC7M [2:0]			OC7PE	OC7FE	Res.	Res.
	Reset value															0	0									0	0	0	0	0	0		
0x07C.. 0x8C	Reserved	Res.																															
0x090	TIM_MPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK4NM[1:0]		BK4M[1:0]		BK3NM[1:0]		BK3M[1:0]		BK2NM[1:0]		BK2M[1:0]		BK1NM[1:0]		BK1M[1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 391. TIM1/TIM8/TIM20 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x094	TIM_MPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x098.. 0x09C	Reserved	Res.																															
0x0A0	TIM_OOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	OOS4N	OOS4	OOS3N	OOS3	OOS2N	OOS2	OOS1N	OOS1
	Reset value																				0					0	0	0	0	0	0	0	0
0x0A4.. 0x3D8	Reserved	Res.																															
0x3DC	TIM_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	TIM_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

35 General-purpose timers (TIM2/TIM3/TIM4/TIM5)

35.1 TIM2/TIM3/TIM4/TIM5 introduction

The general-purpose timers consist of a 16-bit or 32-bit autoreload counter driven by a programmable prescaler.

They can be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 35.4.27: Timer synchronization*.

35.2 TIM2/TIM3/TIM4/TIM5 main features

General-purpose TIMx timer features include:

- 16-bit or 32-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to four independent channels for:
  - Input capture.
  - Output compare.
  - PWM generation (edge- and center-aligned modes).
  - One-pulse mode output.
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger).
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger).
  - Input capture.
  - Output compare.
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes.
- Trigger input for external clock or cycle-by-cycle current management.
- ADC synchronization for jitter-free sampling points.

35.3 TIM2/TIM3/TIM4/TIM5 implementation

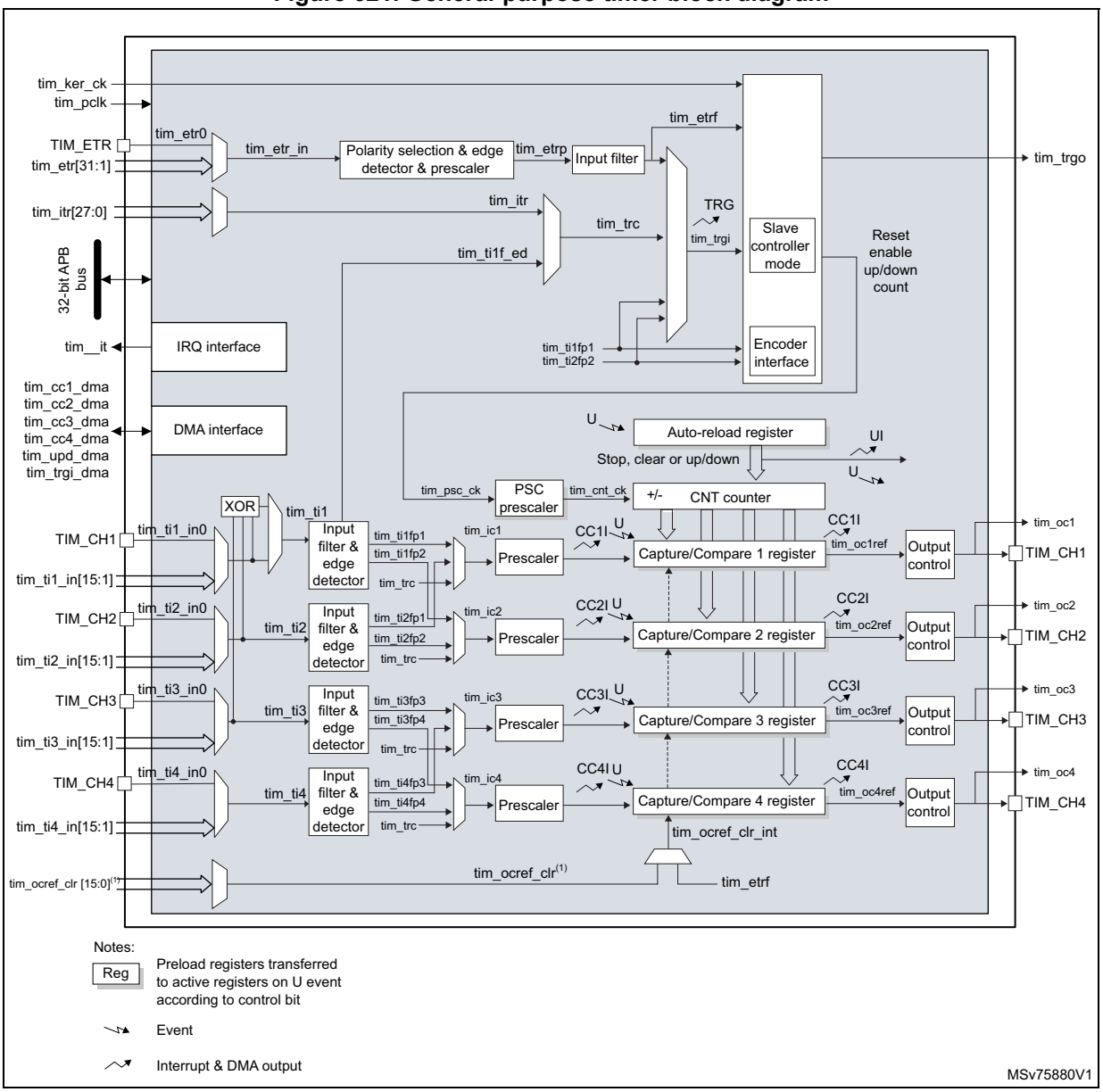
Table 392. STM32H7Pxxx general purpose timers

Timer instance	TIM2	TIM3	TIM4	TIM5
Resolution	32-bit	32-bit	32-bit	32-bit
OCREF clear selection	Yes	Yes	Yes	Yes
Sources	tim_etrf tim_ocref_clr	tim_etrf tim_ocref_clr	tim_etrf tim_ocref_clr	tim_etrf tim_ocref_clr

35.4 TIM2/TIM3/TIM4/TIM5 functional description

35.4.1 Block diagram

Figure 621. General-purpose timer block diagram



1. This feature is not available on all timers, refer to *Section 35.3: TIM2/TIM3/TIM4/TIM5 implementation*.

35.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals

Table 393 and *Table 394* in this section summarize the TIM inputs and outputs.

Table 393. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) , external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.

Table 394. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[31:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulse width control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[27:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[15:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulse width control.
tim_pclk	Input	Timer APB clock.
tim_ker_ck	Input	Timer kernel clock

Table 394. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_it	Output	Global Timer interrupt, gathering capture/compare, update and break trigger requests.
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture/compare [4:1] dma requests.
tim_upd_dma	Output	Timer update dma request.
tim_trgi_dma	Output	Timer trigger dma request.
tim_ocx (x = 1 to 4)	Output	Internal output compare signals of channels 1 to 4. these outputs can be used for triggering/synchronizing other internal peripherals.

Tables below list the sources connected to the tim_ti[4:1] input multiplexers.

Table 395. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti1_in0	TIM2_CH1	TIM3_CH1	TIM4_CH1	TIM5_CH1
tim_ti1_in1	comp1_out	comp1_out	comp1_out	comp1_out
tim_ti1_in2	comp3_out	comp3_out	comp3_out	comp3_out
tim_ti1_in3	comp5_out	comp5_out	comp5_out	comp5_out
tim_ti1_in4	comp7_out	comp7_out	comp7_out	comp7_out
tim_ti1_in5	-	-	-	-
tim_ti1_in6	-	-	-	-
tim_ti1_in7	tim1_trgo	tim1_trgo	tim1_trgo	tim1_trgo
tim_ti1_in8	tim15_trgo	tim2_trgo	tim2_trgo	tim2_trgo
tim_ti1_in9	-	-	-	-
tim_ti1_in10	fdcan1_rxeof_evt	fdcan2_rxeof_evt	fdcan3_rxeof_evt	fdcan1_rxeof_evt
tim_ti1_in[15:11]	-	-	-	-

Table 396. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti2_in0	TIM2_CH2	TIM3_CH2	TIM4_CH2	TIM5_CH2
tim_ti2_in1	comp2_out	comp2_out	comp2_out	comp2_out

Table 396. Interconnect to the tim_ti2 input multiplexer (continued)

tim_ti2 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti2_in2	comp4_out	comp4_out	comp4_out	comp4_out
tim_ti2_in3	comp6_out	comp6_out	comp6_out	comp6_out
tim_ti2_in4	comp8_out	comp8_out	comp8_out	comp8_out
tim_ti2_in5	-	-	-	-
tim_ti2_in6	-	-	-	-
tim_ti2_in7	tim3_trgo	tim15_trgo	tim3_trgo	tim3_trgo
tim_ti2_in8	tim8_trgo	tim8_trgo	tim8_trgo	tim8_trgo
tim_ti2_in9	-	-	-	-
tim_ti2_in10	fdcan1_txeof_evt	fdcan2_txeof_evt	fdcan3_txeof_evt	fdcan1_txeof_evt
tim_ti2_in[15:11]	-	-	-	-

Table 397. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti3_in0	TIM2_CH3	TIM3_CH3	TIM4_CH3	TIM5_CH3
tim_ti3_in1	tim4_trgo	tim4_trgo	tim15_trgo	tim4_trgo
tim_ti3_in2	tim20_trgo	tim20_trgo	tim20_trgo	tim20_trgo
tim_ti3_in3	-	-	-	-
tim_ti3_in4	fdcan2_rxeof_evt	fdcan3_rxeof_evt	fdcan1_rxeof_evt	fdcan2_rxeof_evt
tim_ti3_in[15:5]	-	-	-	-

Table 398. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti4_in0	TIM2_CH4	TIM3_CH4	TIM4_CH4	TIM5_CH4
tim_ti4_in1	tim5_trgo	tim5_trgo	tim5_trgo	tim15_trgo
tim_ti4_in2	-	-	-	-
tim_ti4_in3	-	-	-	-
tim_ti4_in4	fdcan2_txeof_evt	fdcan3_txeof_evt	fdcan1_txeof_evt	fdcan2_txeof_evt
tim_ti4_in[15:5]	-	-	-	-

The table below lists the internal sources connected to the tim_itr input multiplexer.

Table 399. TIMx internal trigger connection

TIMx	TIM2	TIM3	TIM4	TIM5
tim_itr0	tim1_trgo	tim1_trgo	tim1_trgo	tim1_trgo
tim_itr1	tim1_trgo2	tim1_trgo2	tim1_trgo2	tim1_trgo2
tim_itr2	-	tim2_trgo	tim2_trgo	tim2_trgo
tim_itr3	tim3_trgo	-	tim3_trgo	tim3_trgo
tim_itr4	tim4_trgo	tim4_trgo	-	tim4_trgo
tim_itr5	tim5_trgo	tim5_trgo	tim5_trgo	-
tim_itr6	tim8_trgo	tim8_trgo	tim8_trgo	tim8_trgo
tim_itr7	tim8_trgo2	tim8_trgo2	tim8_trgo2	tim8_trgo2
tim_itr8	tim9_trgo	tim9_trgo	tim9_trgo	tim9_trgo
tim_itr9	tim10_oc1	tim10_oc1	tim10_oc1	tim10_oc1
tim_itr10	tim11_oc1	tim11_oc1	tim11_oc1	tim11_oc1
tim_itr11	tim12_trgo	tim12_trgo	tim12_trgo	tim12_trgo
tim_itr12	tim13_oc1	tim13_oc1	tim13_oc1	tim13_oc1
tim_itr13	tim14_oc1	tim14_oc1	tim14_oc1	tim14_oc1
tim_itr14	tim15_trgo	tim15_trgo	tim15_trgo	tim15_trgo
tim_itr15	tim16_oc1	tim16_oc1	tim16_oc1	tim16_oc1
tim_itr16	tim17_oc1	tim17_oc1	tim17_oc1	tim17_oc1
tim_itr17	tim20_trgo	tim20_trgo	tim20_trgo	tim20_trgo
tim_itr18	tim20_trgo2	tim20_trgo2	tim20_trgo2	tim20_trgo2
tim_itr19	otgfs_sof	-	-	otgfs_sof
tim_itr20	-	-	-	-
tim_itr21	hrtim1_sync_out2	hrtim1_sync_out2	hrtim1_sync_out2	hrtim1_sync_out2
tim_itr22	hrtim2_sync_out2	hrtim2_sync_out2	hrtim2_sync_out2	hrtim2_sync_out2
tim_itr[27:23]	-	-	-	-

The table below lists the internal sources connected to the tim_etr input multiplexer.

Table 400. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signal assignment			
	TIM2	TIM3	TIM4	TIM5
tim_etr0	TIM2_ETR	TIM3_ETR	TIM4_ETR	TIM5_ETR
tim_etr1	comp1_out	comp1_out	comp1_out	comp1_out
tim_etr2	comp2_out	comp2_out	comp2_out	comp2_out
tim_etr3	comp3_out	comp3_out	comp3_out	comp3_out

Table 400. Interconnect to the tim_etr input multiplexer (continued)

Timer external trigger input signal	Timer external trigger signal assignment			
	TIM2	TIM3	TIM4	TIM5
tim_etr4	comp4_out	comp4_out	comp4_out	comp4_out
tim_etr5	comp5_out	comp5_out	comp5_out	comp5_out
tim_etr6	comp6_out	comp6_out	comp6_out	comp6_out
tim_etr7	comp7_out	comp7_out	comp7_out	comp7_out
tim_etr8	comp8_out	comp8_out	comp8_out	comp8_out
tim_etr9	-	-	-	-
tim_etr10	-	-	-	-
tim_etr11	-	-	-	-
tim_etr12	-	-	-	-
tim_etr13	-	TIM2_ETR	TIM2_ETR	TIM2_ETR
tim_etr14	TIM3_ETR	-	TIM3_ETR	TIM3_ETR
tim_etr15	TIM4_ETR	TIM4_ETR	-	TIM4_ETR
tim_etr16	TIM5_ETR	TIM5_ETR	TIM5_ETR	-
tim_etr17	adc1_awd1	adc2_awd1	adc3_awd1	adc4_awd1
tim_etr18	adc1_awd2	adc2_awd2	adc3_awd2	adc4_awd2
tim_etr19	adc1_awd3	adc2_awd3	adc3_awd3	adc4_awd3
tim_etr20	-	-	-	-
tim_etr21	-	-	-	-
tim_etr22	lse_ck	-	-	-
tim_etr23	-	-	-	-
tim_etr24	-	-	-	-
tim_etr[31:25]	-	-	-	-

Table 401. Interconnect to the tim_ocref_clr input multiplexer

Timer tim_ocref_clr signal	Timer tim_ocref_clr signals assignment			
	TIM2	TIM3	TIM4	TIM5
tim_ocref_clr0	comp1_out	comp1_out	comp1_out	comp1_out
tim_ocref_clr1	comp2_out	comp2_out	comp2_out	comp2_out
tim_ocref_clr2	comp3_out	comp3_out	comp3_out	comp3_out
tim_ocref_clr3	comp4_out	comp4_out	comp4_out	comp4_out
tim_ocref_clr4	comp5_out	comp5_out	comp5_out	comp5_out
tim_ocref_clr5	comp6_out	comp6_out	comp6_out	comp6_out
tim_ocref_clr6	comp7_out	comp7_out	comp7_out	comp7_out

Table 401. Interconnect to the tim_ocref_clr input multiplexer (continued)

Timer tim_ocref_clr signal	Timer tim_ocref_clr signals assignment			
	TIM2	TIM3	TIM4	TIM5
tim_ocref_clr7	comp8_out	comp8_out	comp8_out	comp8_out
tim_ocref_clr[11:8]	-	-	-	-
tim_ocref_clr12	-	-	-	-
tim_ocref_clr13	-	-	-	-
tim_ocref_clr[15:14]	-	-	-	-

35.4.3 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIM_CNT)
- Prescaler register (TIM_PSC)
- Autoreload register (TIM_ARR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIM_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when down-counting) and if the UDIS bit equals 0 in the TIM_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

Note: An update overrun status bit (UIOVRF in TIM_SR register) indicates that an update request occurred while an update interrupt is pending (update interrupt flag set). This bit is useful when debugging closed loop applications, to indicate that:

- the update interrupt service routine is too long with respect to the interval between consecutive update events

- the update interrupt service routine was served too late with respect to the subsequent update event

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIM_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note: The counter starts counting 1 clock cycle after setting the CEN bit in the TIM_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIM_PSC

register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 622 and *Figure 623* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 622. Counter timing diagram with prescaler division change from 1 to 2

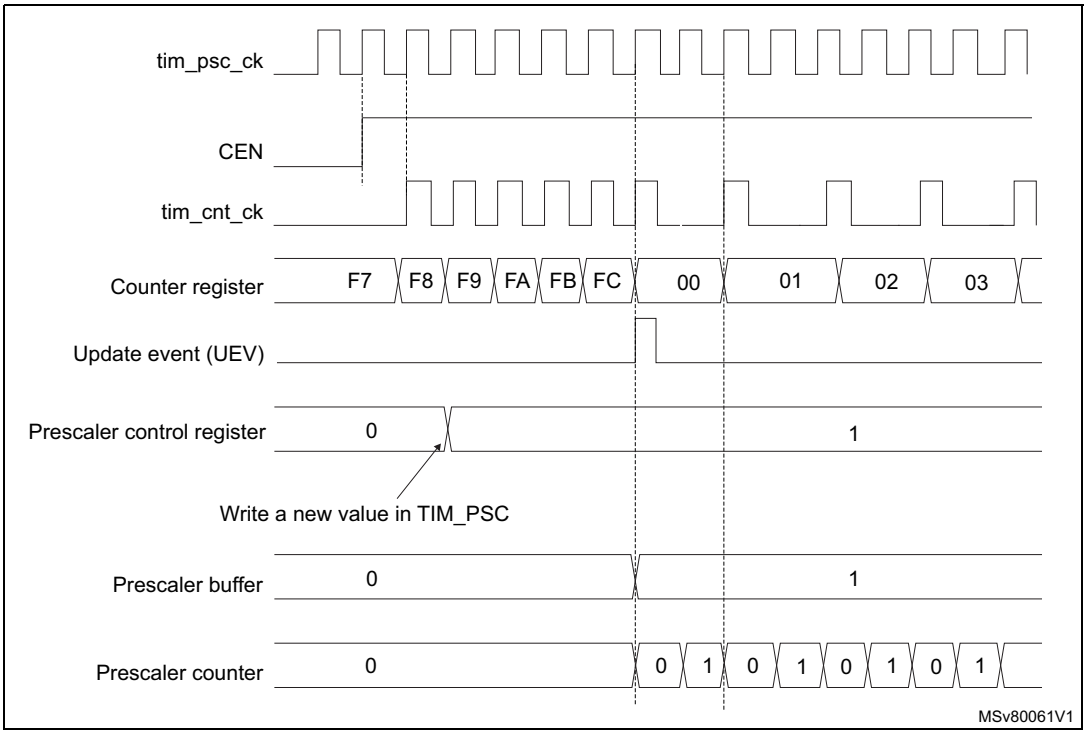
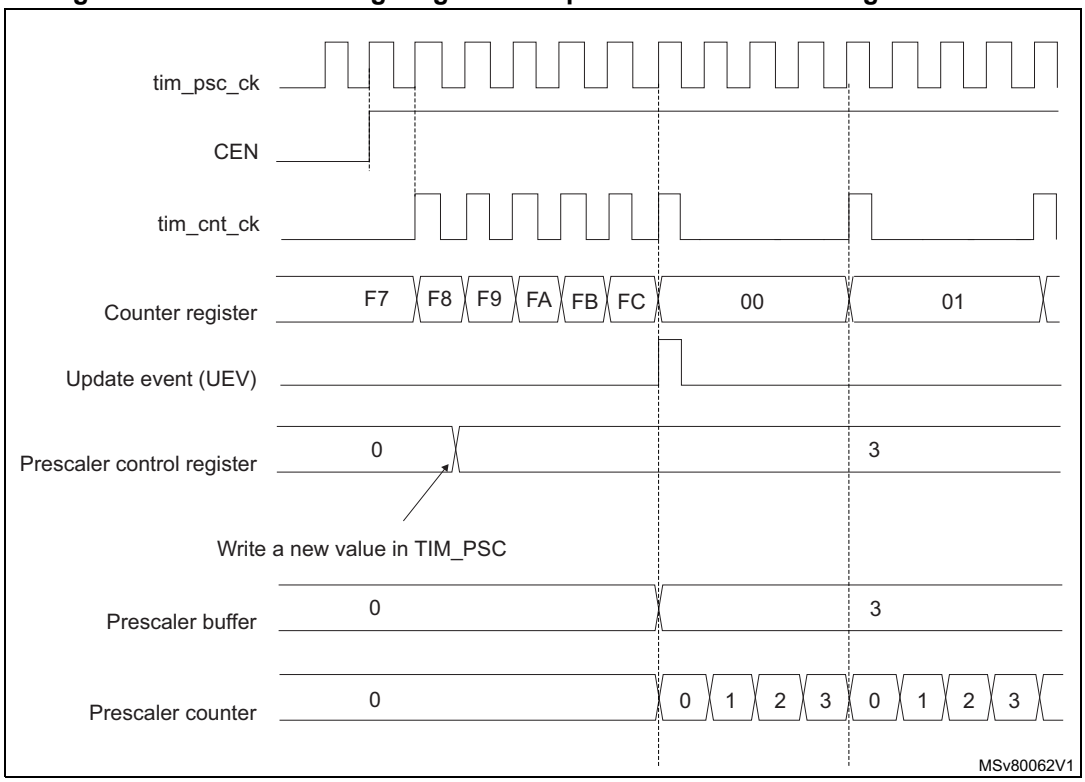


Figure 623. Counter timing diagram with prescaler division change from 1 to 4



35.4.4 Counter modes

Up-counting mode

In up-counting mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).
- The autoreload shadow register is updated with the preload value (TIM_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 624. Counter timing diagram, internal clock divided by 1

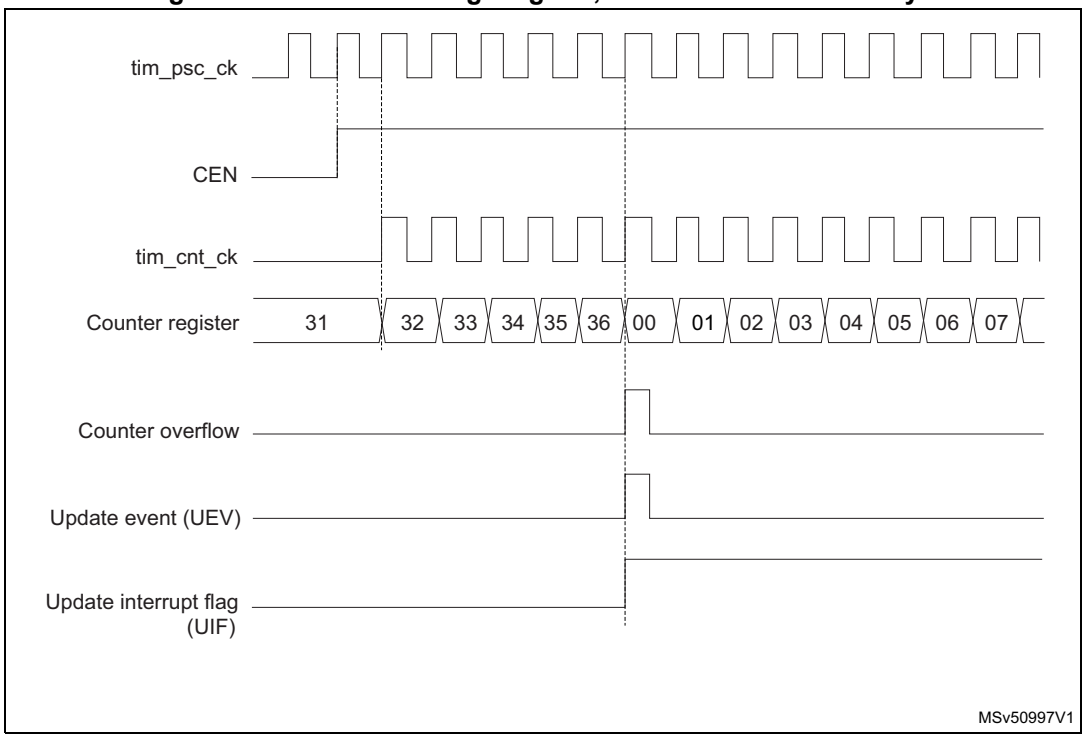


Figure 625. Counter timing diagram, internal clock divided by 2

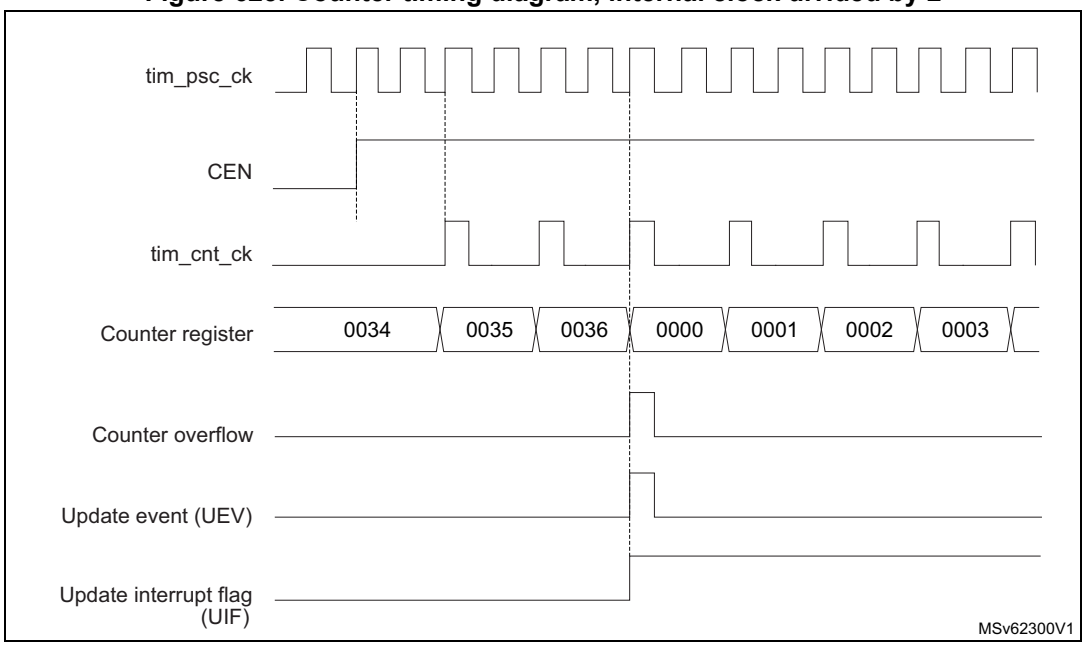


Figure 626. Counter timing diagram, internal clock divided by 4

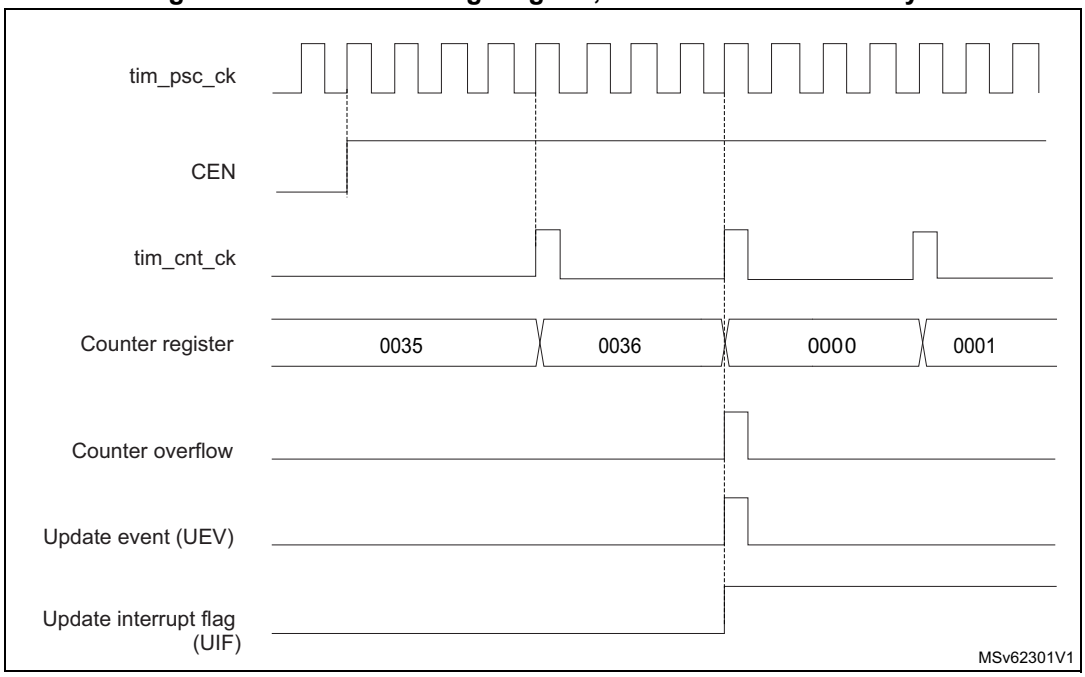


Figure 627. Counter timing diagram, internal clock divided by N

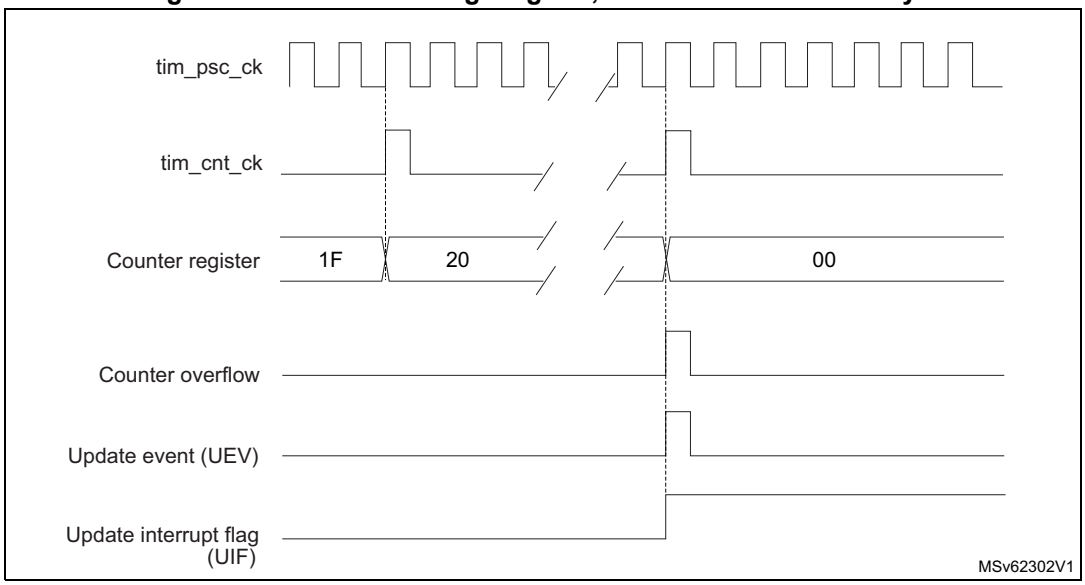


Figure 628. Counter timing diagram, Update event when ARPE = 0 (TIM_ARR not preloaded)

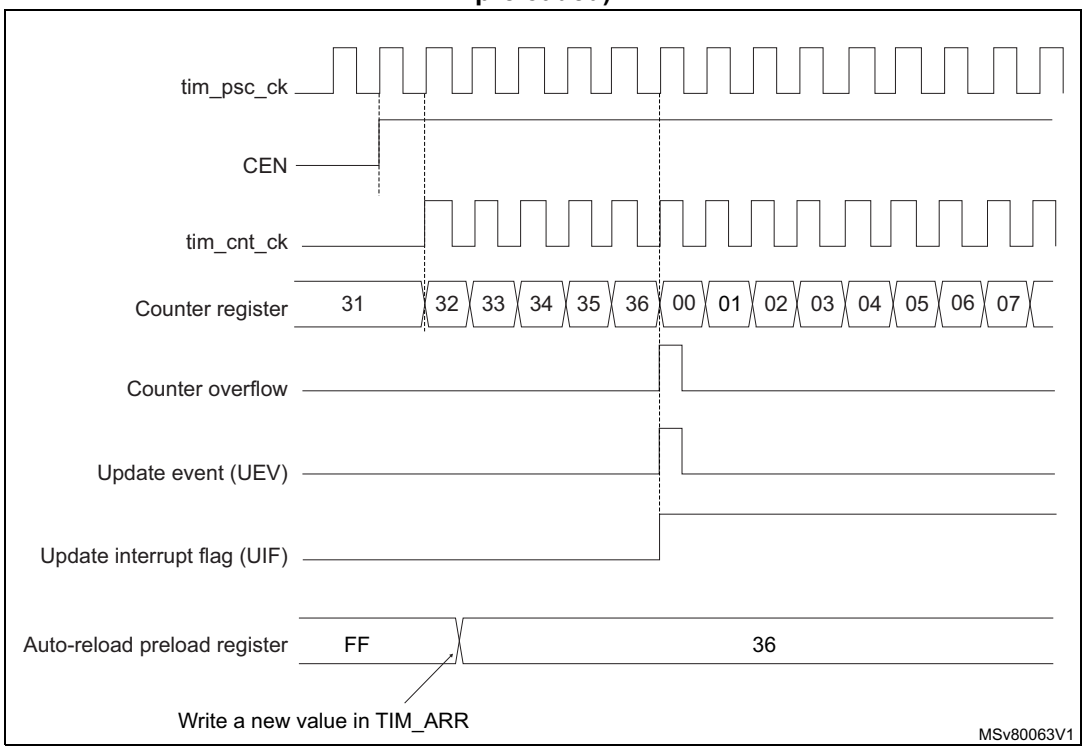
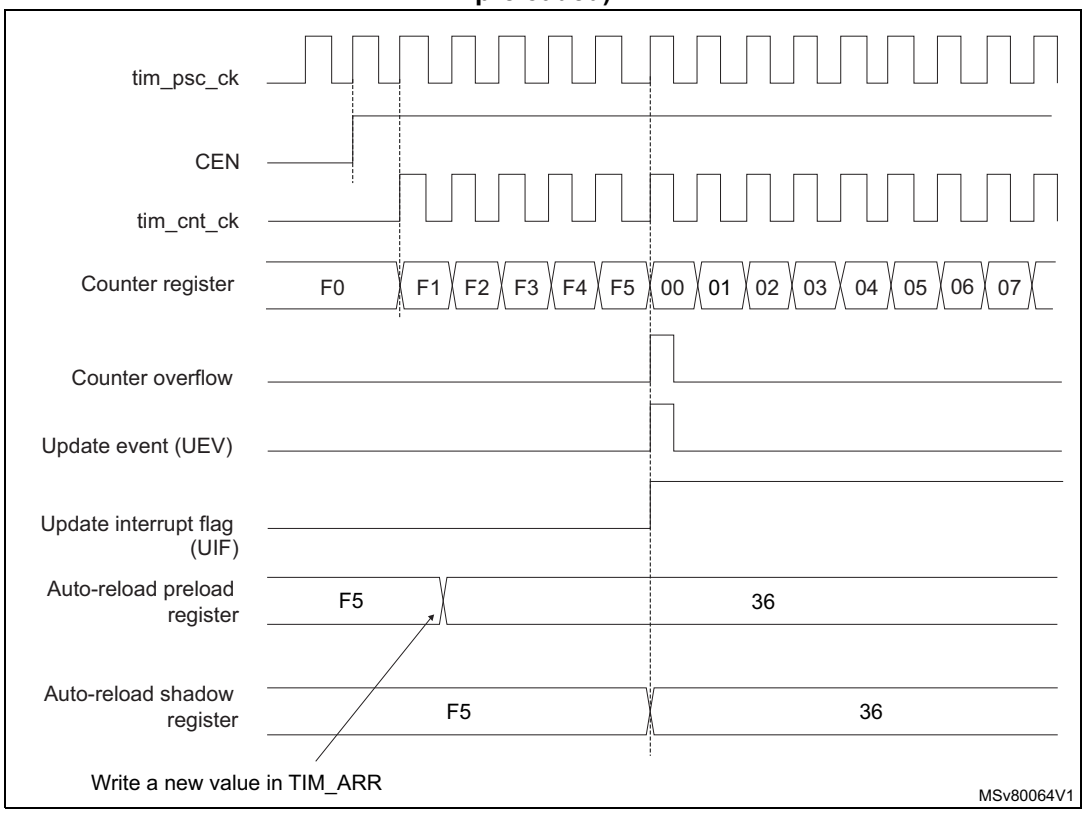


Figure 629. Counter timing diagram, Update event when ARPE = 1 (TIM_ARR preloaded)



Down-counting mode

In down-counting mode, the counter counts from the autoreload value (content of the TIM_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

An update event can be generated at each counter underflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller)

The UEV update event can be disabled by software by setting the UDIS bit in TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate does not change).

In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 630. Counter timing diagram, internal clock divided by 1

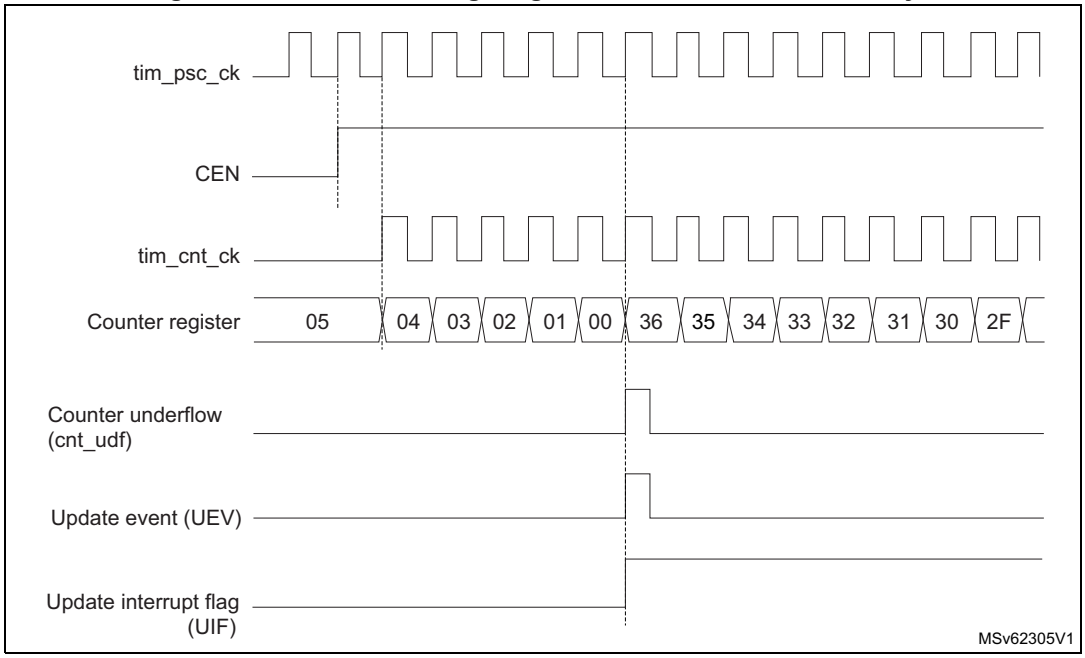


Figure 631. Counter timing diagram, internal clock divided by 2

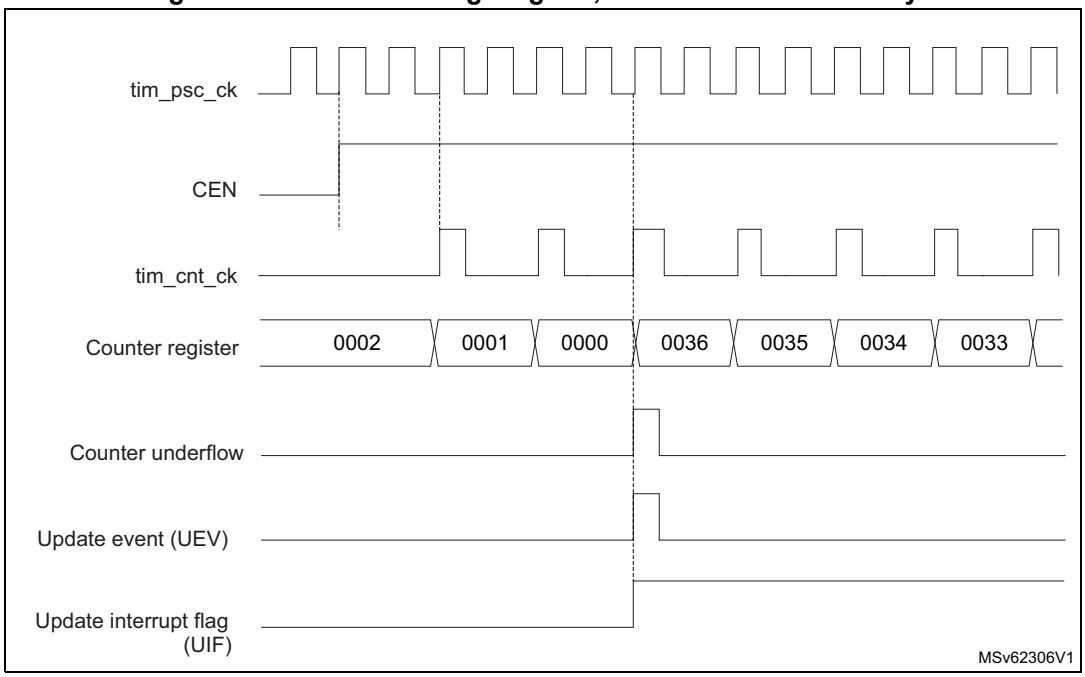


Figure 632. Counter timing diagram, internal clock divided by 4

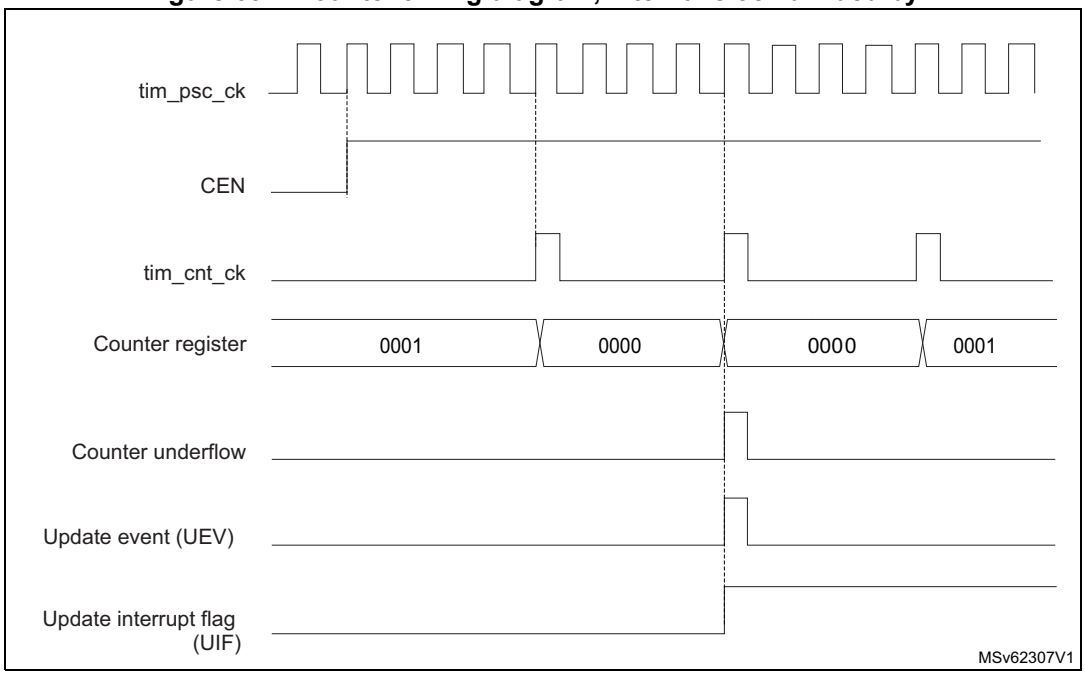


Figure 633. Counter timing diagram, internal clock divided by N

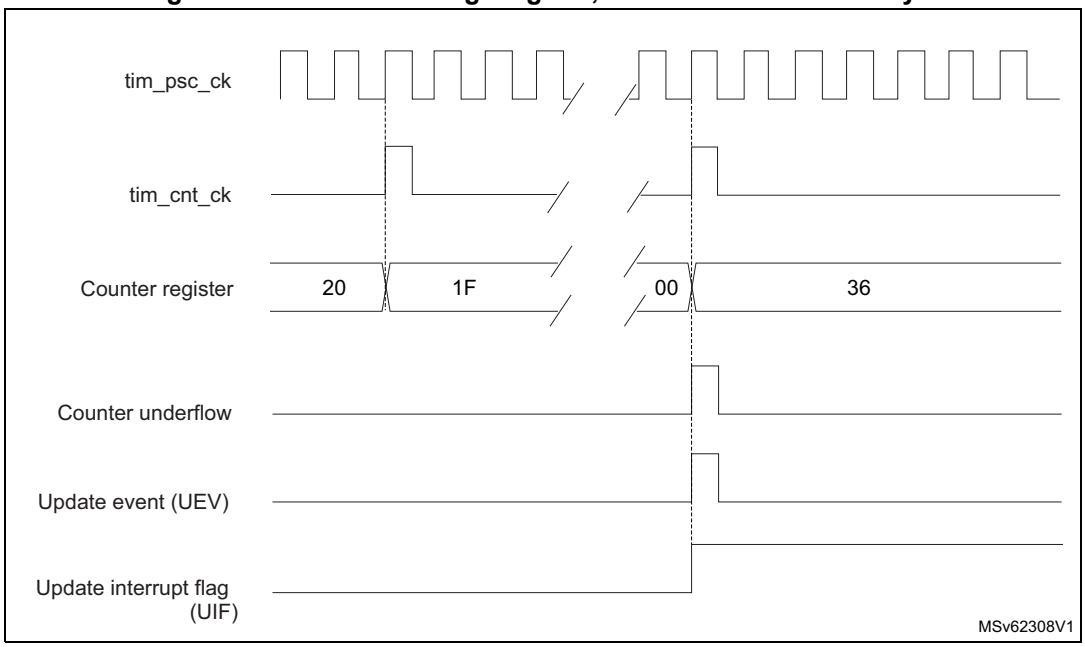
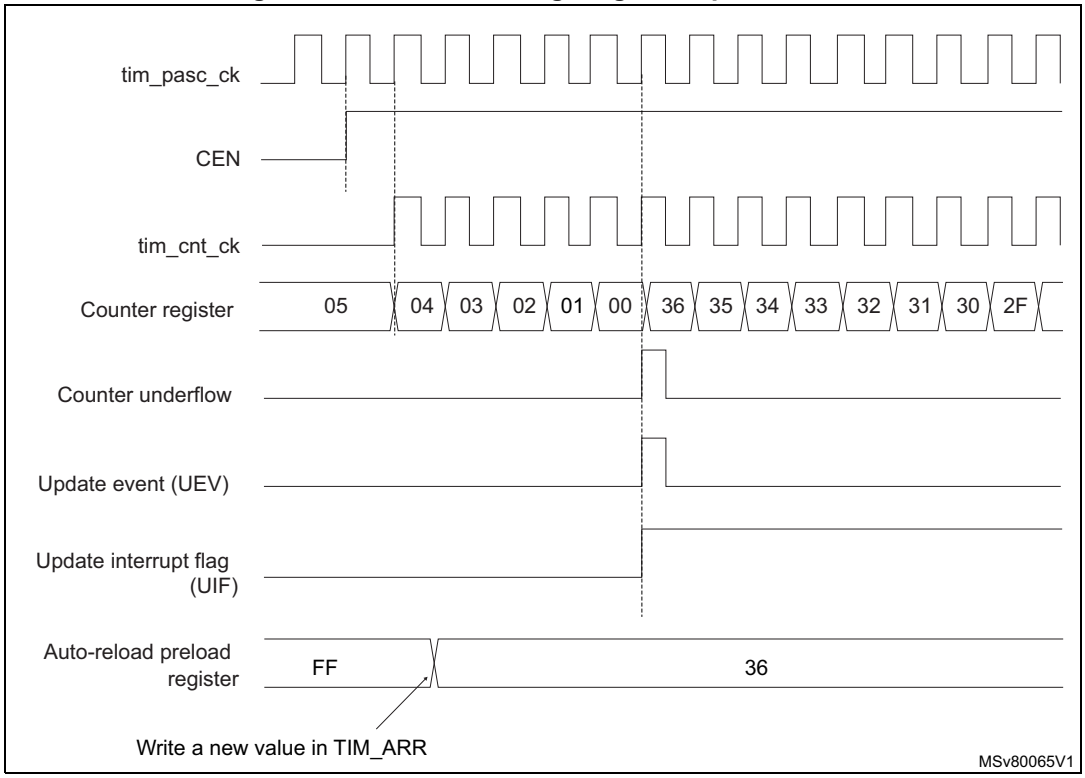


Figure 634. Counter timing diagram, Update event



Center-aligned mode (up/down-counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIM_CR1 register are not equal to 00. The output compare interrupt flag of channels configured in output is set when: the counter counts down (center aligned mode 1, CMS = 01), the counter counts up (center aligned mode 2, CMS = 10) the counter counts up and down (center aligned mode 3, CMS = 11).

In this mode, the direction bit (DIR from TIM_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

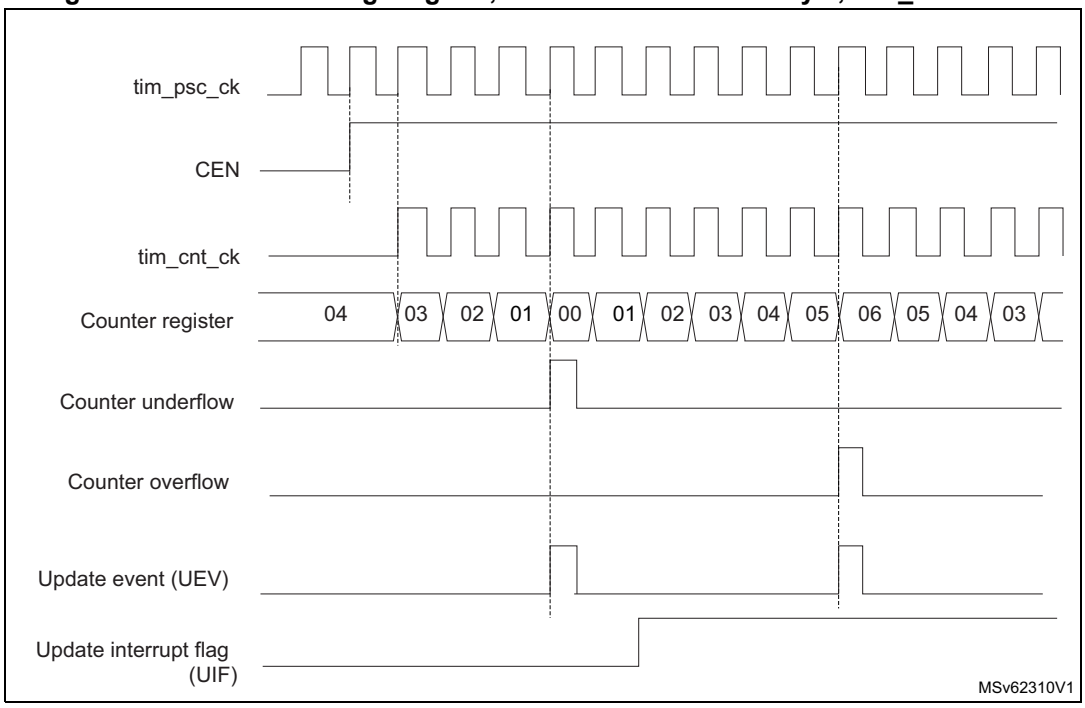
In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 635. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 35.5.1: TIM control register 1 (TIM_CR1)*).

Figure 636. Counter timing diagram, internal clock divided by 2

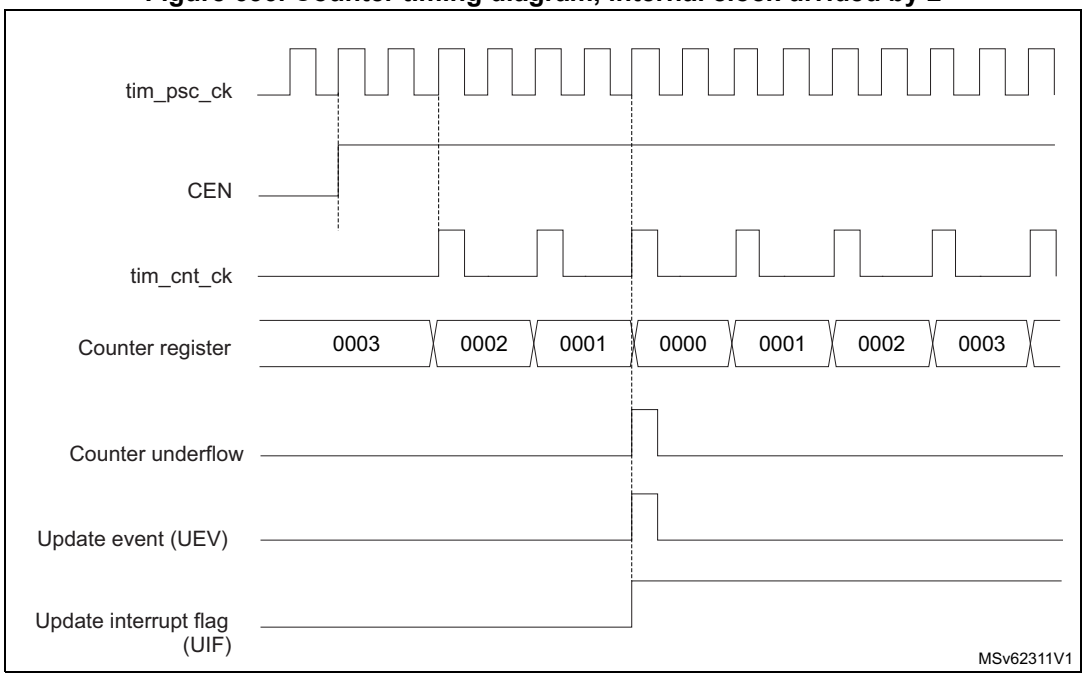
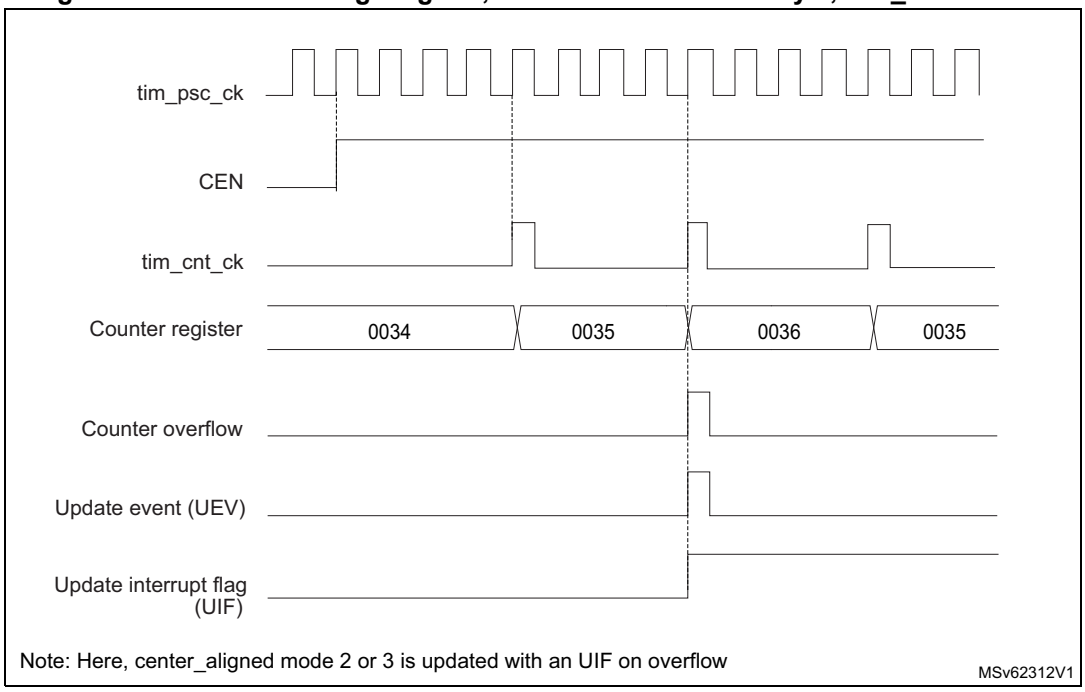


Figure 637. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36



1. Center-aligned mode 2 or 3 is used with a UIF on overflow.

Figure 638. Counter timing diagram, internal clock divided by N

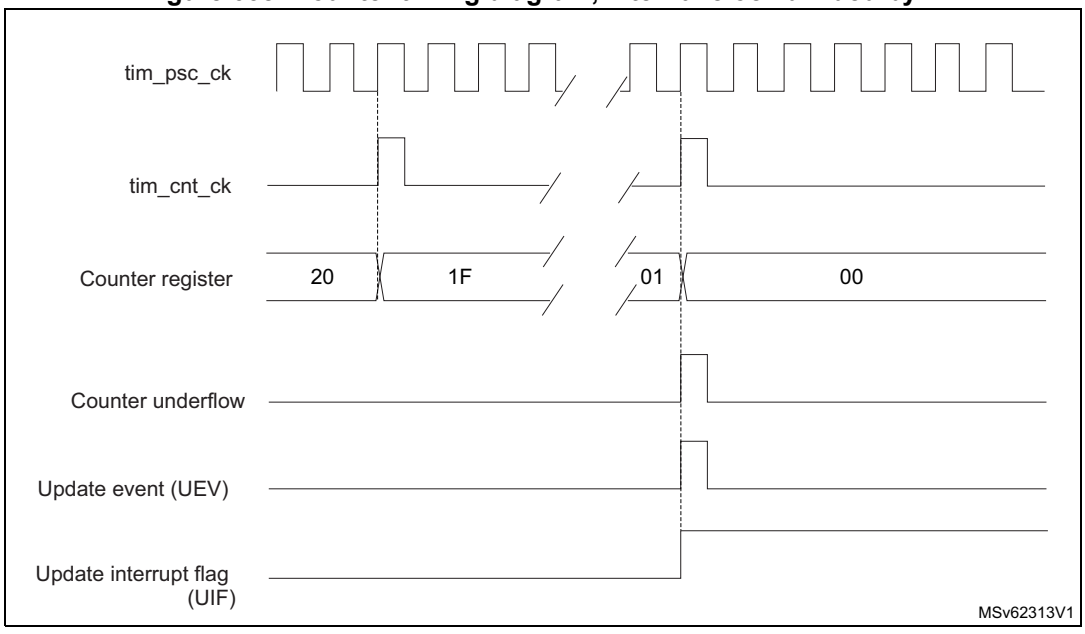


Figure 639. Counter timing diagram, Update event with ARPE = 1 (counter underflow)

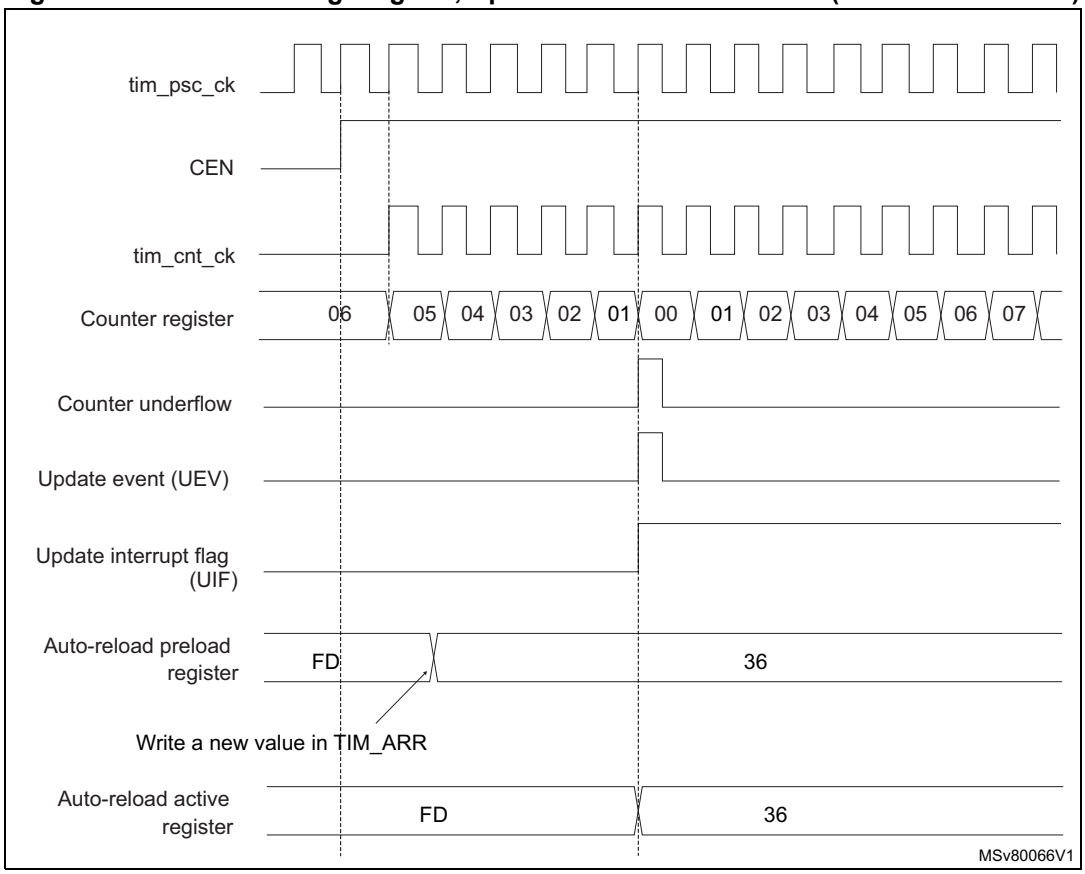
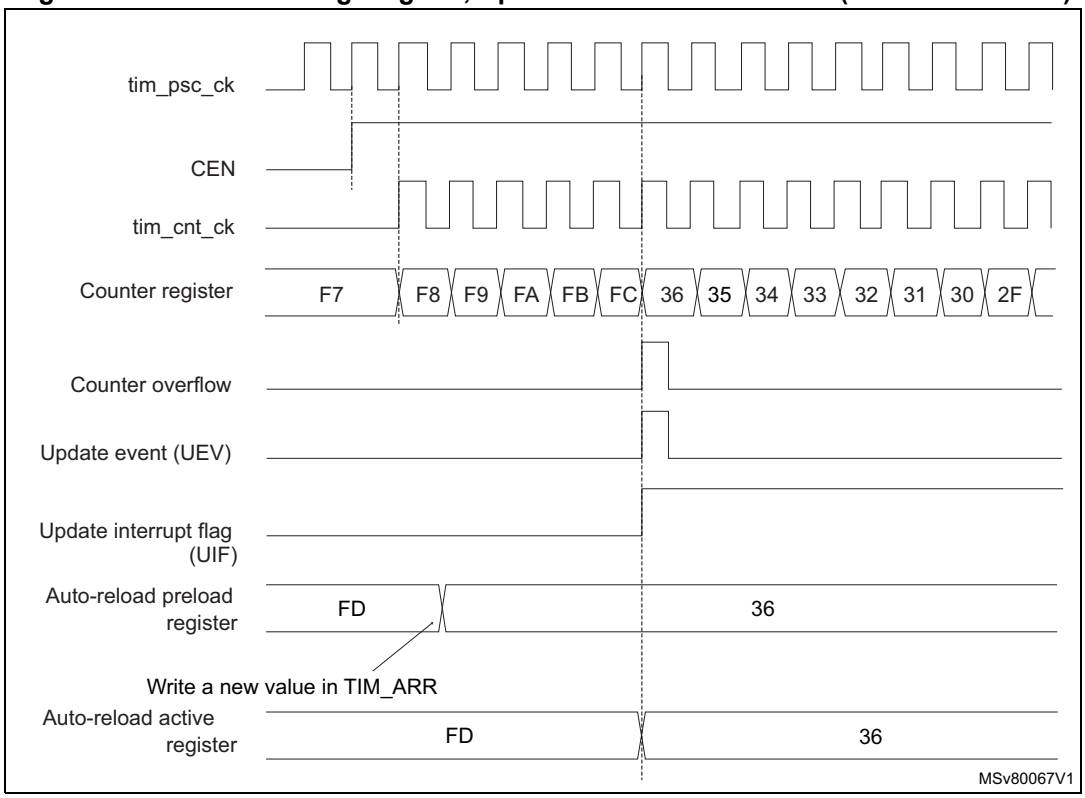


Figure 640. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



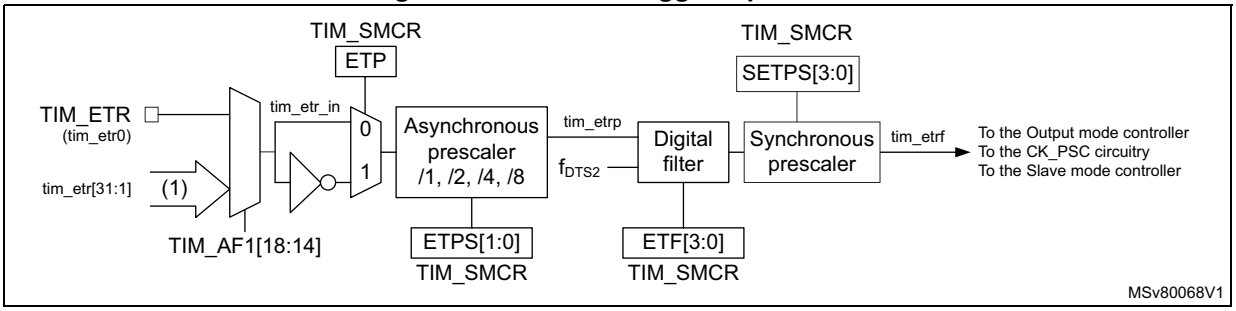
35.4.5 External trigger input

The timer features an external trigger input tim_etr_in. It can be used as:

- external clock (external clock mode 2, see *Section 35.4.6*)
- trigger for the slave mode (see *Section 35.4.27*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 35.4.16*)

Figure 641 below describes the tim_etr_in input conditioning. The input polarity is defined with the ETP bit in TIM_SMCR register. The trigger frequency can be decreased with the prescalers programmed by the ETPS[1:0] and SETPS[3:0] bitfields and digitally filtered as per the ETF[3:0] bitfield. The resulting signal (tim_etrf) is available for three purposes: as an external clock, to condition the output (typically to reset a PWM output for a current limitation), and as a trigger for the slave mode controller.

Figure 641. External trigger input block



The external trigger input block features 2 prescalers with different functionalities:

- An asynchronous prescaler (division by 1, 2, 4 or 8), able to process tim_etr_in signals with frequencies above tim_ker_ck
- A linear synchronous prescaler (division by 1,2,3,..15), only able to operate with tim_etrp signal frequencies below tim_ker_ck/3

Note: When tim_etrf is used as PWM reset input for cycle-by-cycle current regulation (see Section 35.4.16), SETPS[3:0] must be set to 0 (no prescaling).

Note: The SETPS[3:0] bitfield is always preloaded: a new value will become active only once an update event has occured.

The tim_etr_in input comes from multiple sources: input pins (default configuration), or internal sources. The selection is done with the ETRSEL[4:0] bitfield in the TIM_AF1 register.

Refer to *Section 35.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for the list of sources connected to the etr_in input in the product.

35.4.6 Clock selection

The counter clock can be provided by the following clock sources:

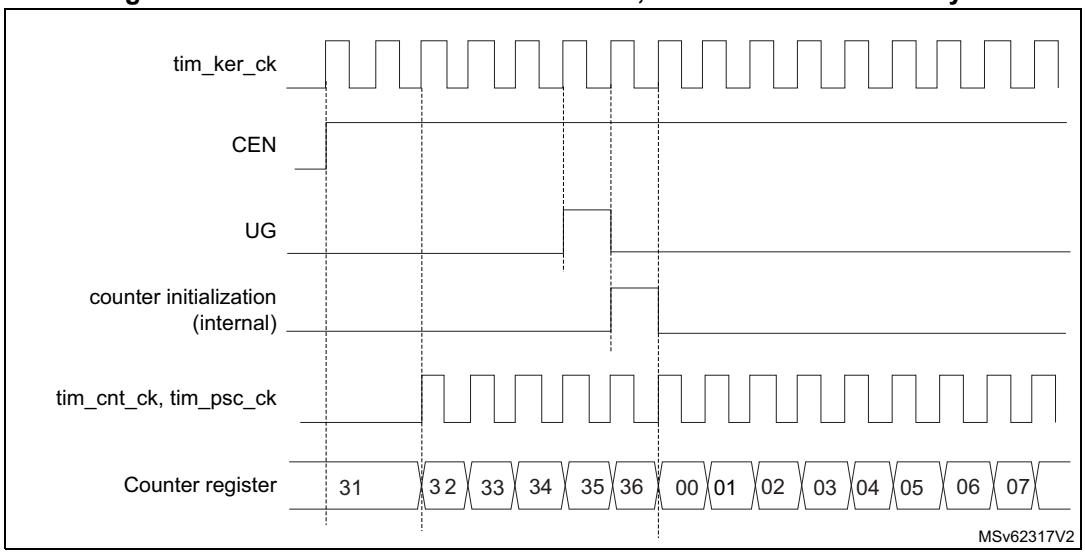
- Internal clock (tim_ker_ck).
- External clock mode1: external input pin (tim_ti1 or tim_ti2).
- External clock mode2: external trigger input (tim_etr_in).
- Internal trigger inputs (tim_itr): using one timer as prescaler for another timer, for example, timer 1 can be configured to act as a prescaler for timer 2. Refer to *Using one timer as prescaler for another timer* for more details.

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000 in the TIM_SMCR register), then the CEN, DIR (in the TIM_CR1 register), and UG bits (in the TIM_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 642 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

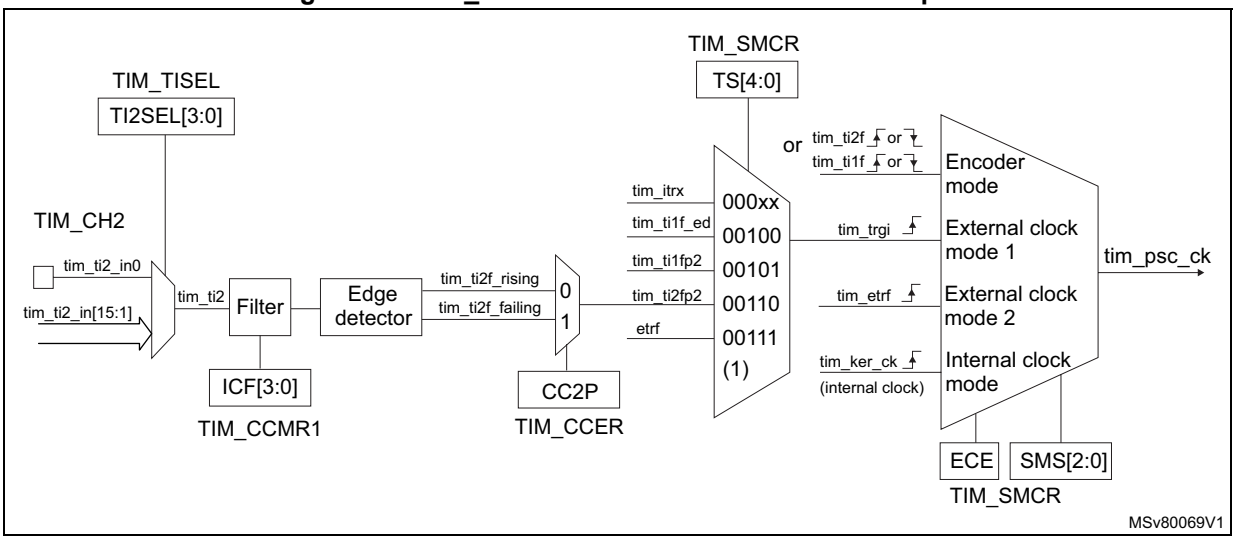
Figure 642. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIM_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 643. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S= 01 in the TIM_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIM_CCMR1 register (if no filter is needed, keep IC2F = 0000).

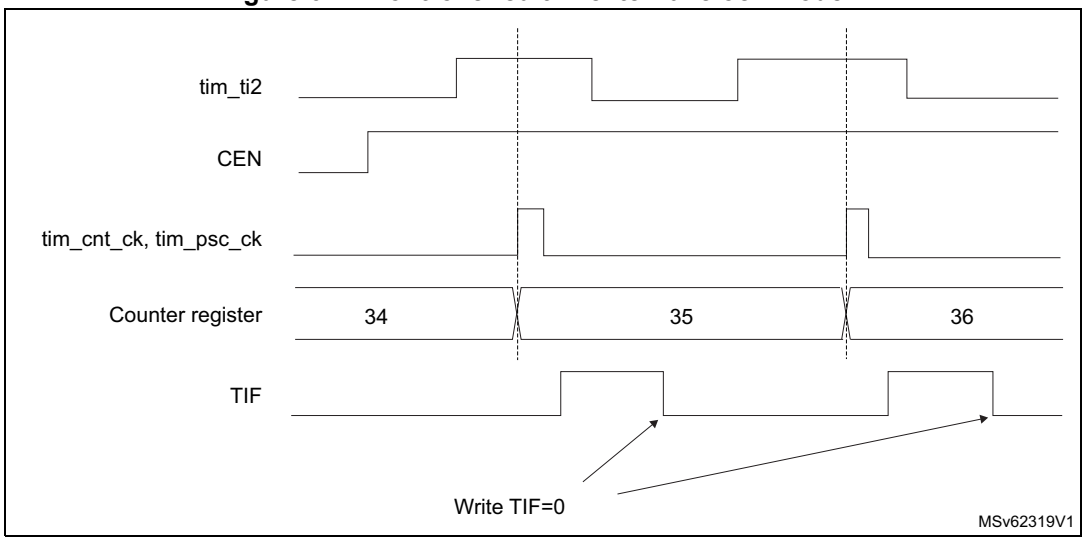
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIM_SMCR register.
6. Select tim_ti2 as the input source by writing TS = 00110 in the TIM_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 644. Control circuit in external clock mode 1



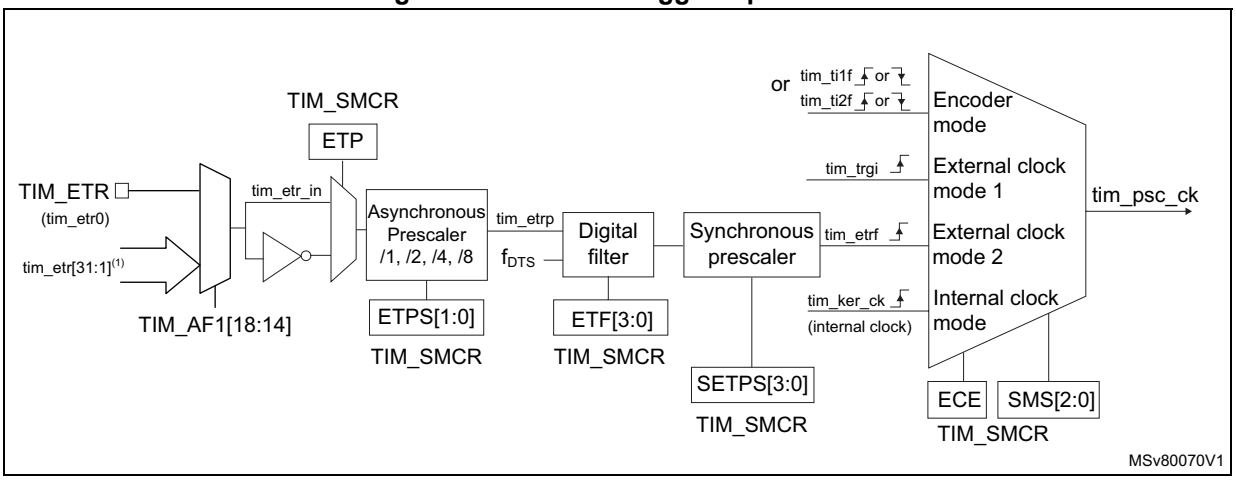
External clock source mode 2

This mode is selected by writing ECE = 1 in the TIM_SMCR register.

The counter can count at each rising or falling edge on the external trigger input tim_etr_in.

Figure 645 gives an overview of the external trigger input block.

Figure 645. External trigger input block



1. Refer to *Section 35.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals*

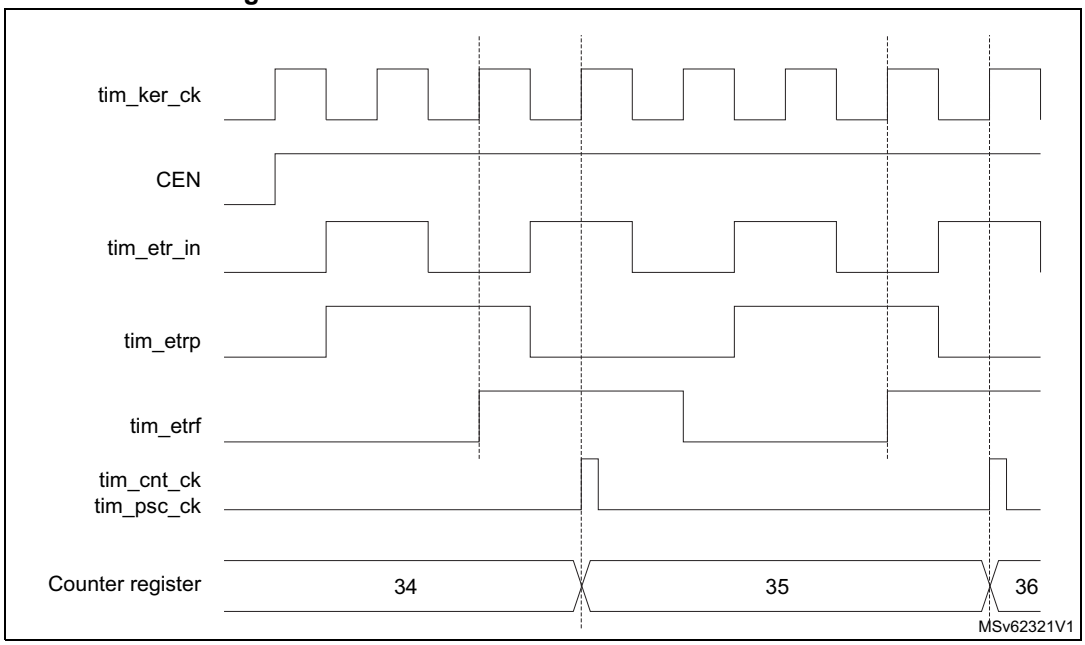
For example, to configure the upcounter to count each two rising edges on tim_etr_in, use the following procedure:

1. Select the proper tim_etr_in source (internal or external) with the ETRSEL[4:0] bits in the TIM_AF1 register.
2. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIM_SMCR register.
3. Set the prescaler by writing ETPS[1:0] = 01 and SETPS[3:0] = 0 in the TIM_SMCR register.
4. Select rising edge detection on the tim_etr_in by writing ETP = 0 in the TIM_SMCR register.
5. Enable external clock mode 2 by writing ECE = 1 in the TIM_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

The counter counts once each two tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency that can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 646. Control circuit in external clock mode 2



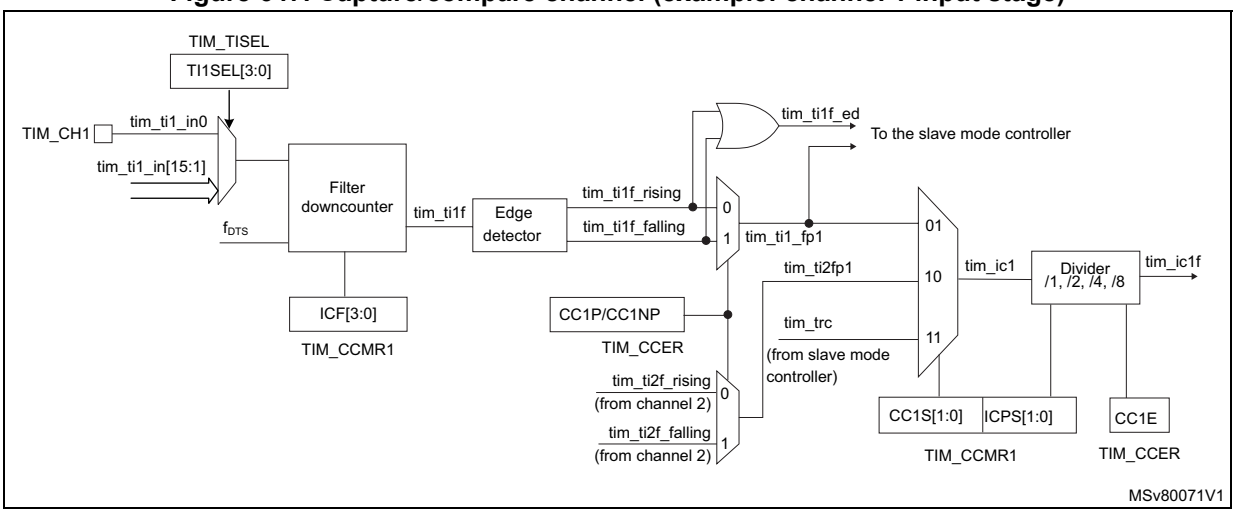
35.4.7 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 647. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 648. Capture/compare channel 1 main circuit

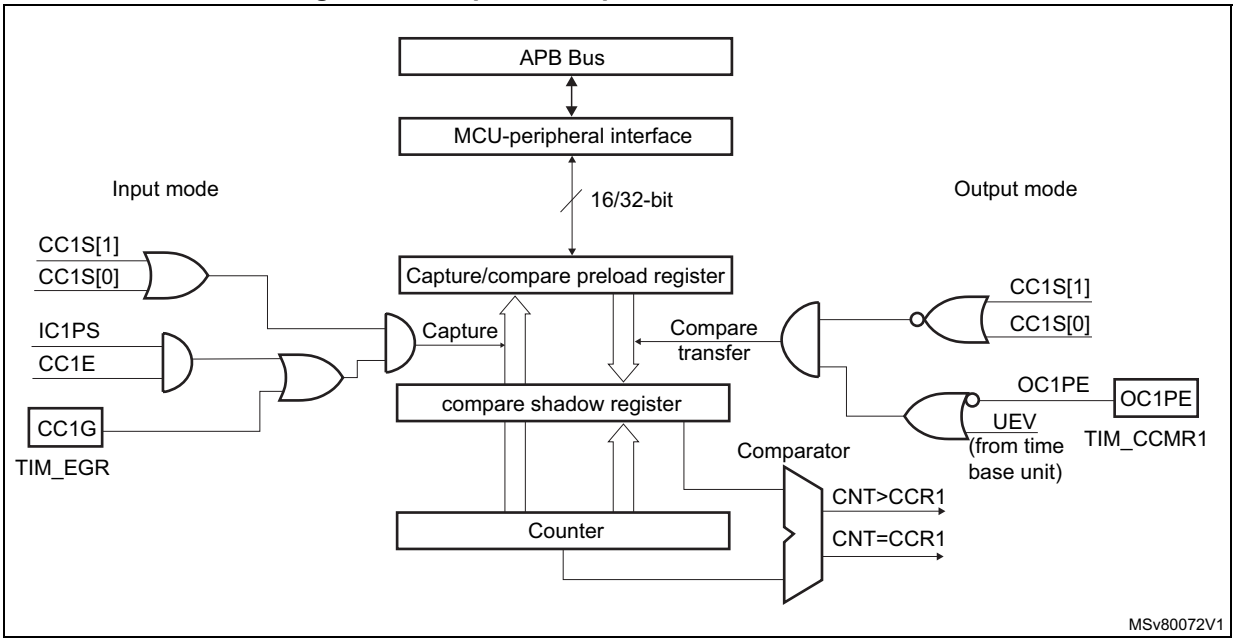
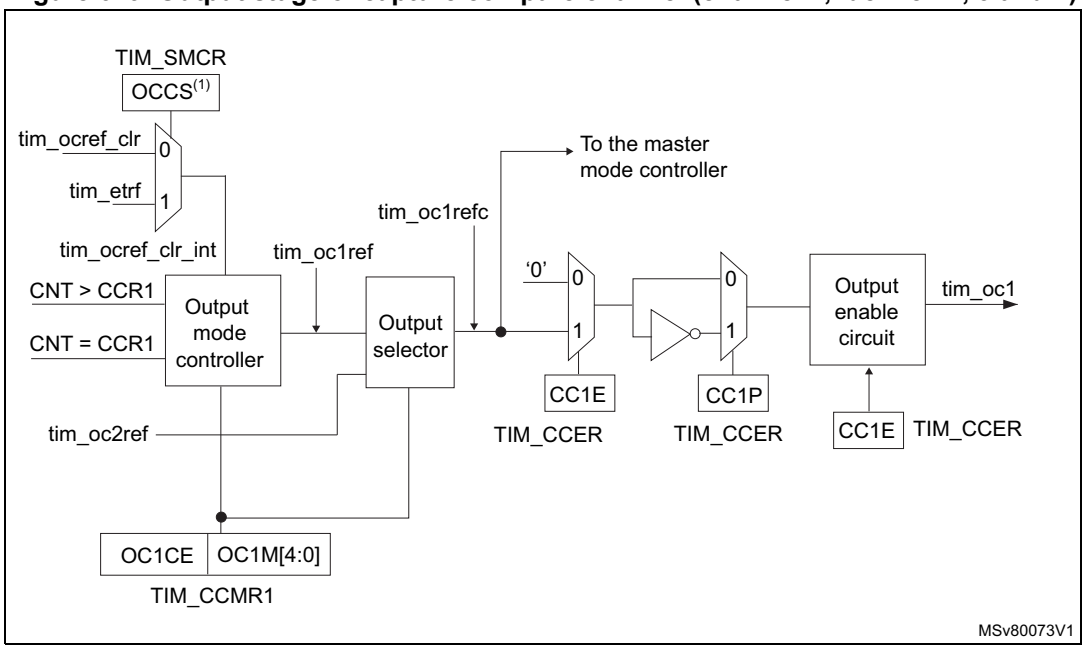


Figure 649. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4)



1. Available on some instances only. If not available, tim_etrf is directly connected to tim_ocref_clr_int.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

35.4.8 Input capture mode

In input capture mode, the capture/compare registers (TIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIM_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIM_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIM_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIM_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input: TIM_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIM_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIM_CCR1 register becomes read-only.
3. Program the needed input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIM_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIM_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing the CC1P and CC1NP bits to 000 in the TIM_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the capture is to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIM_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIM_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIM_DIER register, and/or the DMA request by setting the CC1DE bit in the TIM_DIER register.

When an input capture occurs:

- The TIM_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIM_EGR register.

35.4.9 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIM_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges).

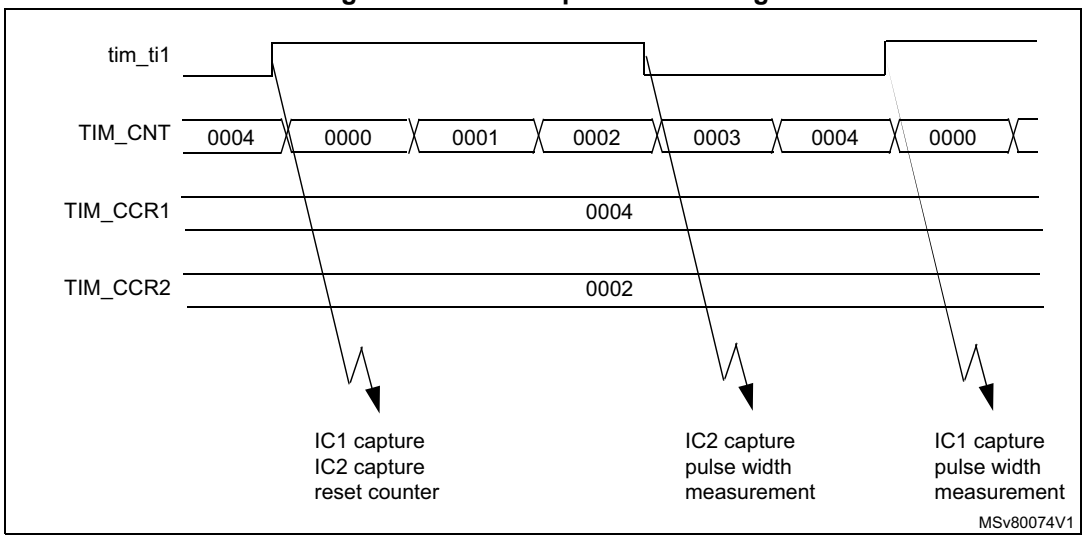
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tix input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulse width of a PWM signal applied on tim_ti1 can be measured using the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input for TIM_CCR1: write the CC1S bits to 01 in the TIM_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIM_CCR1 and counter clear): write the CC1P to 0 and the CC1NP bit to 0 (active on rising edge).
4. Select the active input for TIM_CCR2: write the CC2S bits to 10 in the TIM_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIM_CCR2): write the CC2P bit to 1 and the CC2NP bit to 0 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIM_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIM_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIM_CCER register.

Figure 650. PWM input mode timing



1. The PWM input mode can be used only with the TIM_CH1/TIM_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

35.4.10 Forced output mode

In output mode (CCxS bits = 00 in the TIM_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, the user just needs to write 00101 in the OCxM bits in the corresponding TIM_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

tim_ocxref signal can be forced low by writing the OCxM bits to 00100 in the TIM_CCMRx register.

Anyway, the comparison between the TIM_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section.

35.4.11 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIM_CCMRx register) and the output polarity (CCxP bit in the TIM_CCER register). The output pin can keep its level (OCxM = 00000), be

set active (OCxM = 00001), be set inactive (OCxM = 00010) or can toggle (OCxM = 00011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIM_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIM_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIM_DIER register, CCDS bit in the TIM_CR2 register for the DMA request selection).

The TIM_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIM_CCMRx register.

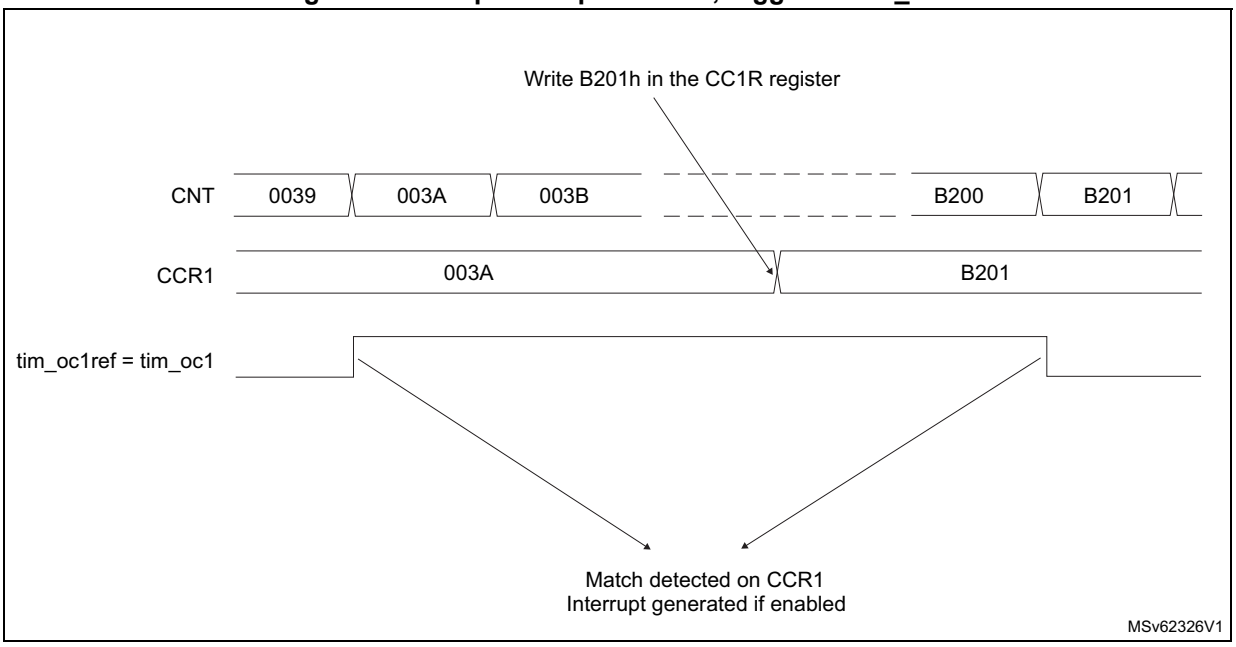
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in one-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIM_ARR and TIM_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example:
   a) Write OCxM = 00011 to toggle tim_ocx output pin when CNT matches CCRx.
   b) Write OCxPE = 0 to disable preload register.
   c) Write CCxP = 0 to select active high polarity.
   d) Write CCxE = 1 to enable the output.
5. Enable the counter by setting the CEN bit in the TIM_CR1 register.

The TIM_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIM_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 651*.

Figure 651. Output compare mode, toggle on tim_oc1



35.4.12 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIM_ARR register and a duty cycle determined by the value of the TIM_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 00110 (PWM mode 1) or 00111 (PWM mode 2) in the OCxM bits in the TIM_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIM_CCMRx register, and eventually the autoreload preload register (in up-counting or center-aligned modes) by setting the ARPE bit in the TIM_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIM_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIM_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by the CCxE bit in the TIM_CCER register. Refer to the TIM_CCERx register description for more details.

In PWM mode (1 or 2), TIM_CNT and TIM_CCRx are always compared to determine whether TIM_CCRx ≤ TIM_CNT or TIM_CNT ≤ TIM_CCRx (depending on the direction of the counter). The tim_ocref_clr can be cleared by an external event through the tim_etr_in or the tim_oceref_clr signals. In this case the tim_ocref_clr signal is asserted only:

- After a compare match event.
- When the output compare mode (OCxM bits in TIM_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM = 00000) to one of the PWM modes (OCxM = 00110 or 00111). This forces the PWM by software while the timer is running.

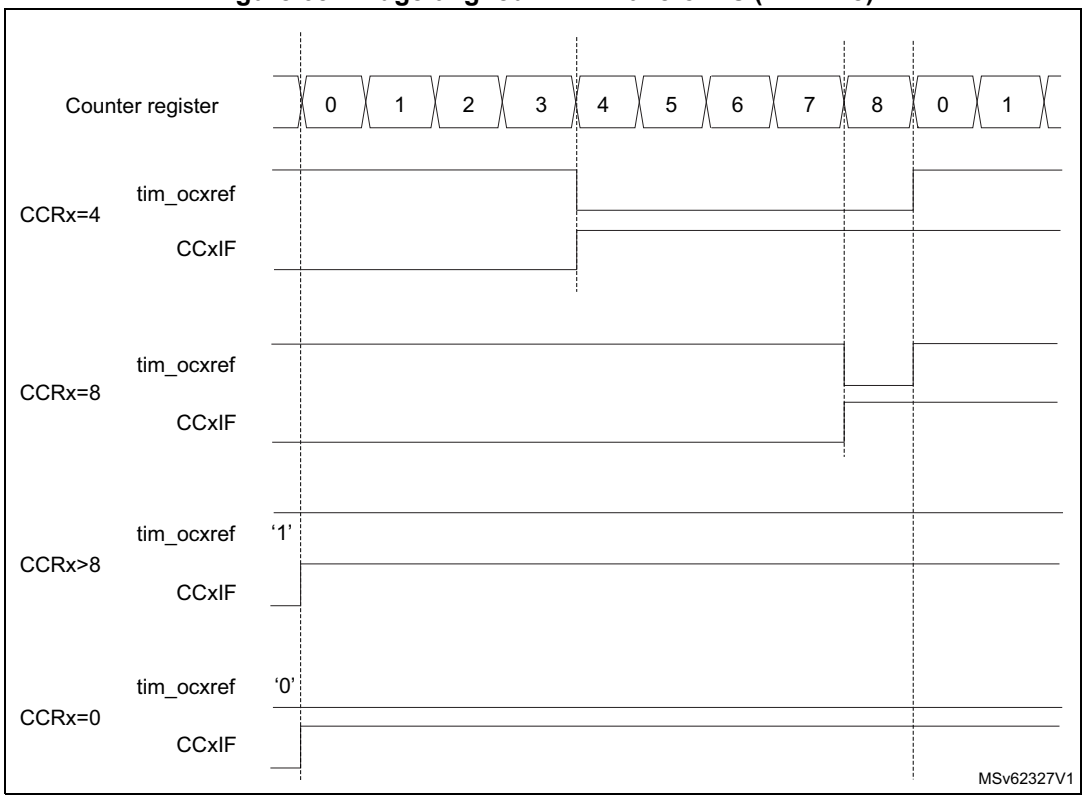
The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIM_CR1 register.

PWM edge-aligned mode

- Up-counting configuration
- Up-counting is active when the DIR bit in the TIM_CR1 register is low. Refer to *Up-counting mode*.

  In the following example, we consider PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIM_CNT <TIM_CCRx else it becomes low. If the compare value in TIM_CCRx is greater than the autoreload value (in TIM_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 652* shows some edge-aligned PWM waveforms in an example where TIM_ARR = 8.

Figure 652. Edge-aligned PWM waveforms (ARR = 8)



Down-counting configuration

- Down-counting is active when DIR bit in TIM_CR1 register is high. Refer to *Down-counting mode*.

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIM_CNT>TIM_CCRx else it becomes high. If the compare value in TIM_CCRx is greater than the autoreload value in TIM_ARR, then tim_ocxref is held at 100%. PWM is not possible in this mode.

PWM center-aligned mode

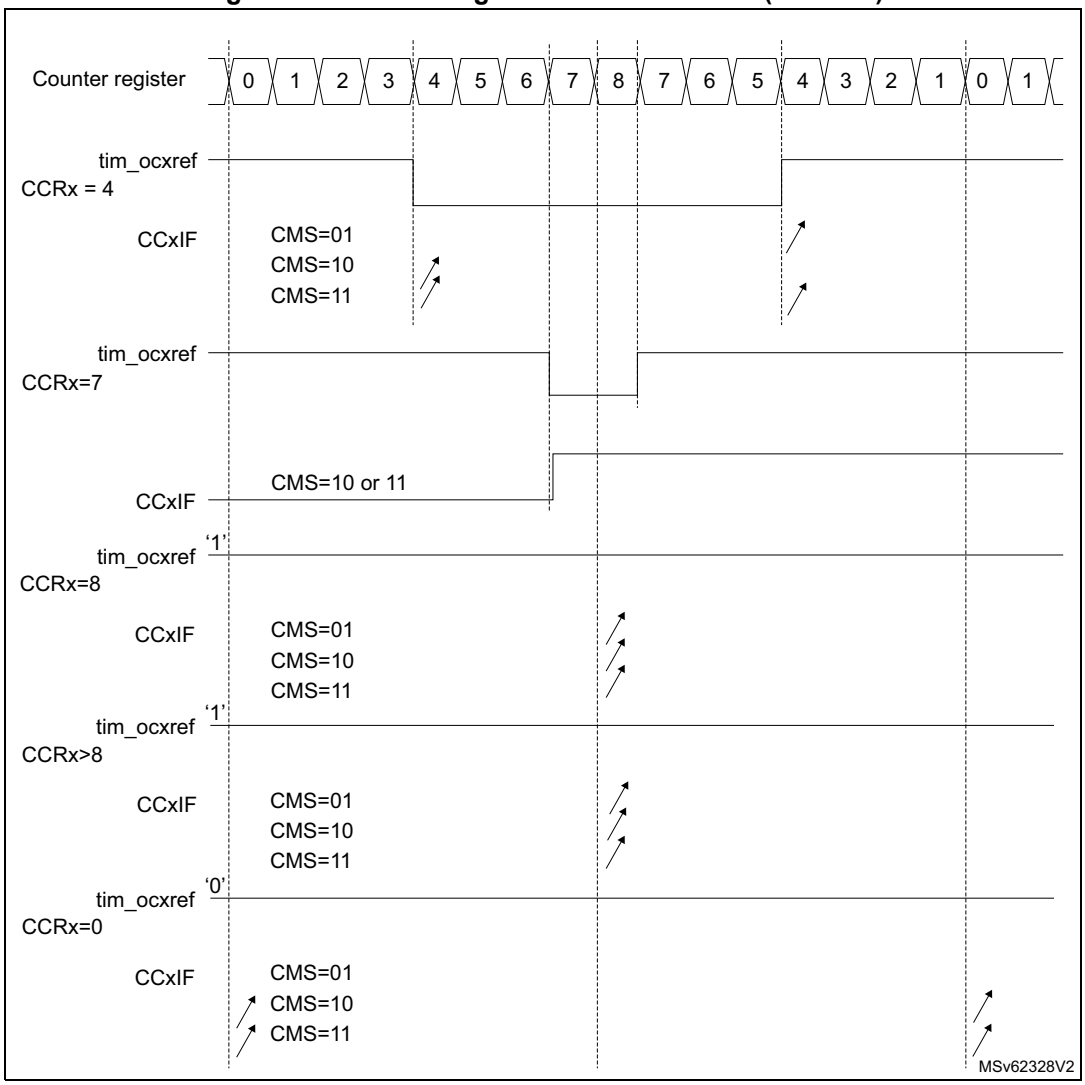
Center-aligned mode is active when the CMS bits in TIM_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the

TIM_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down-counting)*.

Figure 653 shows some center-aligned PWM waveforms in an example where:

- TIM_ARR = 8.
- PWM mode is the PWM mode 1.
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIM_CR1 register.

Figure 653. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIM_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

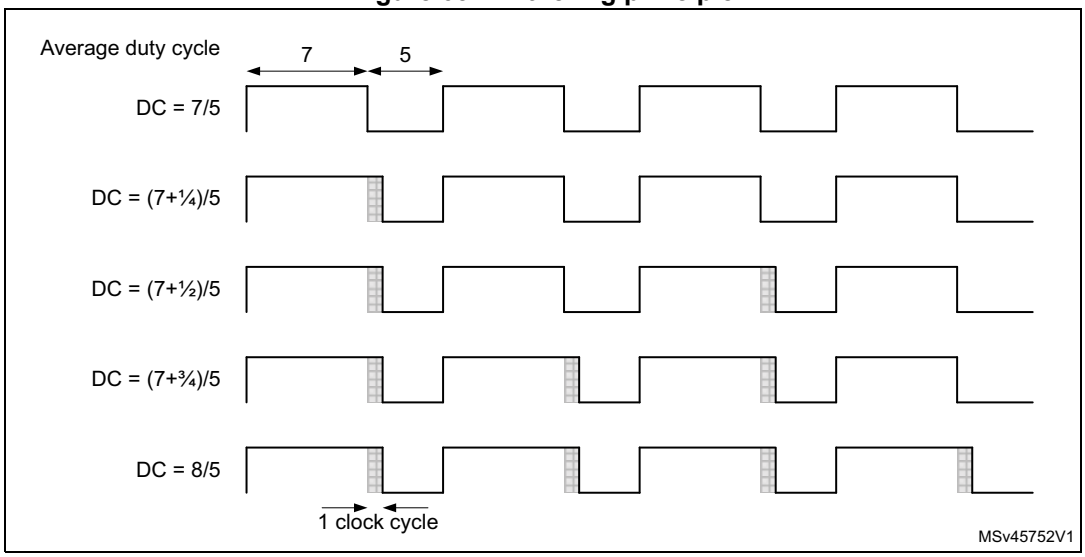
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIM_CNT>TIM_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIM_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIM_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIM_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 654* presents the dithering principle applied to four consecutive PWM cycles.

Figure 654. Dithering principle



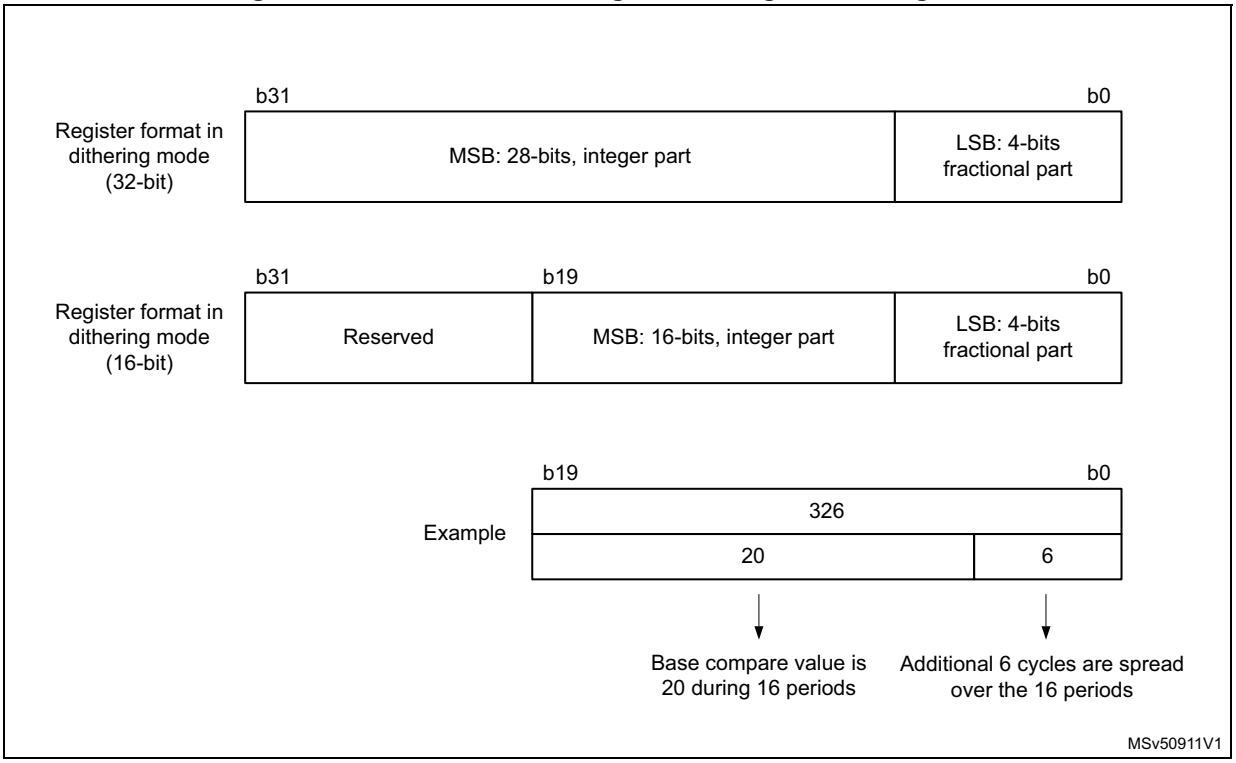
When the dithering mode is enabled, the register coding is changed as following (see *Figure 655* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted by four places and are coding for the base value. In 16-bit mode, the 16-bit format is maintained.

Note: The following sequence must be followed when resetting the DITHEN bit:

1. *CEN and ARPE bits must be reset.*
2. *The DITHEN bit must be reset.*
3. *The CCIF flags must be cleared.*
4. *The CEN bit can be set (eventually with ARPE = 1).*

Figure 655. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode (16-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

$$\text{Dithering mode (32-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{268435454 + \frac{15}{16}}$$

Note: For 16-bit timers, the maximum TIM_ARR and TIM_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part). For 32-bit timers, the maximum TIM_ARR and TIM_CCRy values are limited to

0xFFFFFFEF in dithering mode (corresponds to 264435454 for the integer part and 15 for the dithered part).

As shown on *Figure 656* and *Figure 657*, the dithering mode is used to increase the PWM resolution.

Figure 656. PWM resolution vs frequency (16-bit mode)

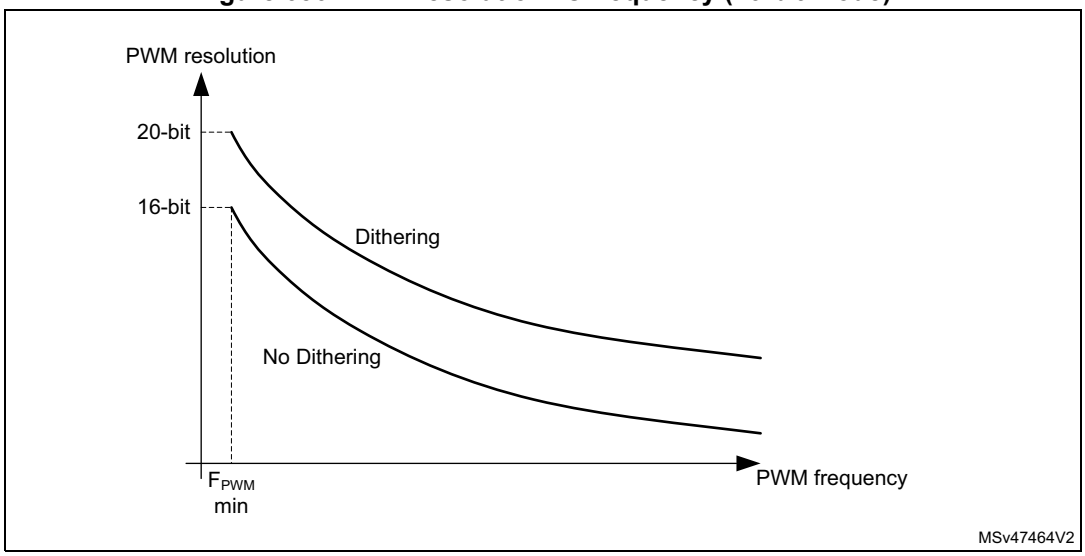
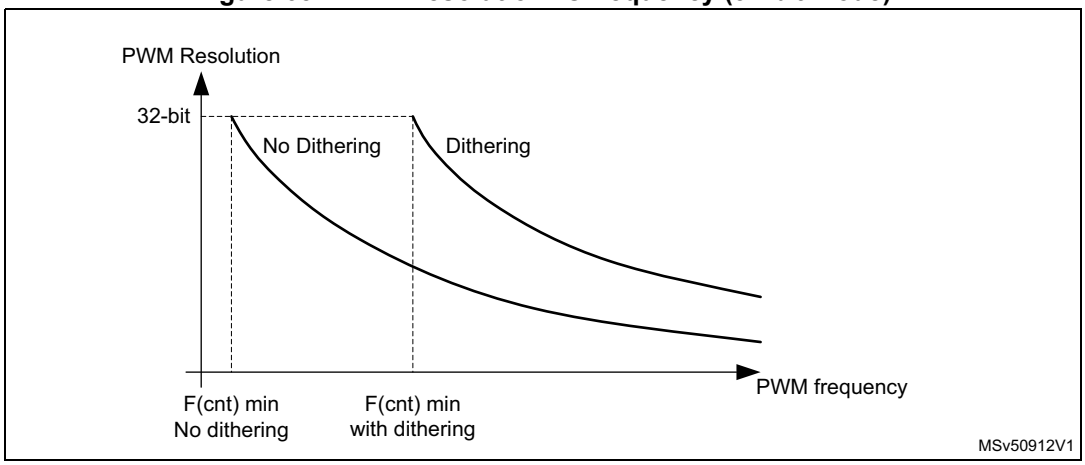
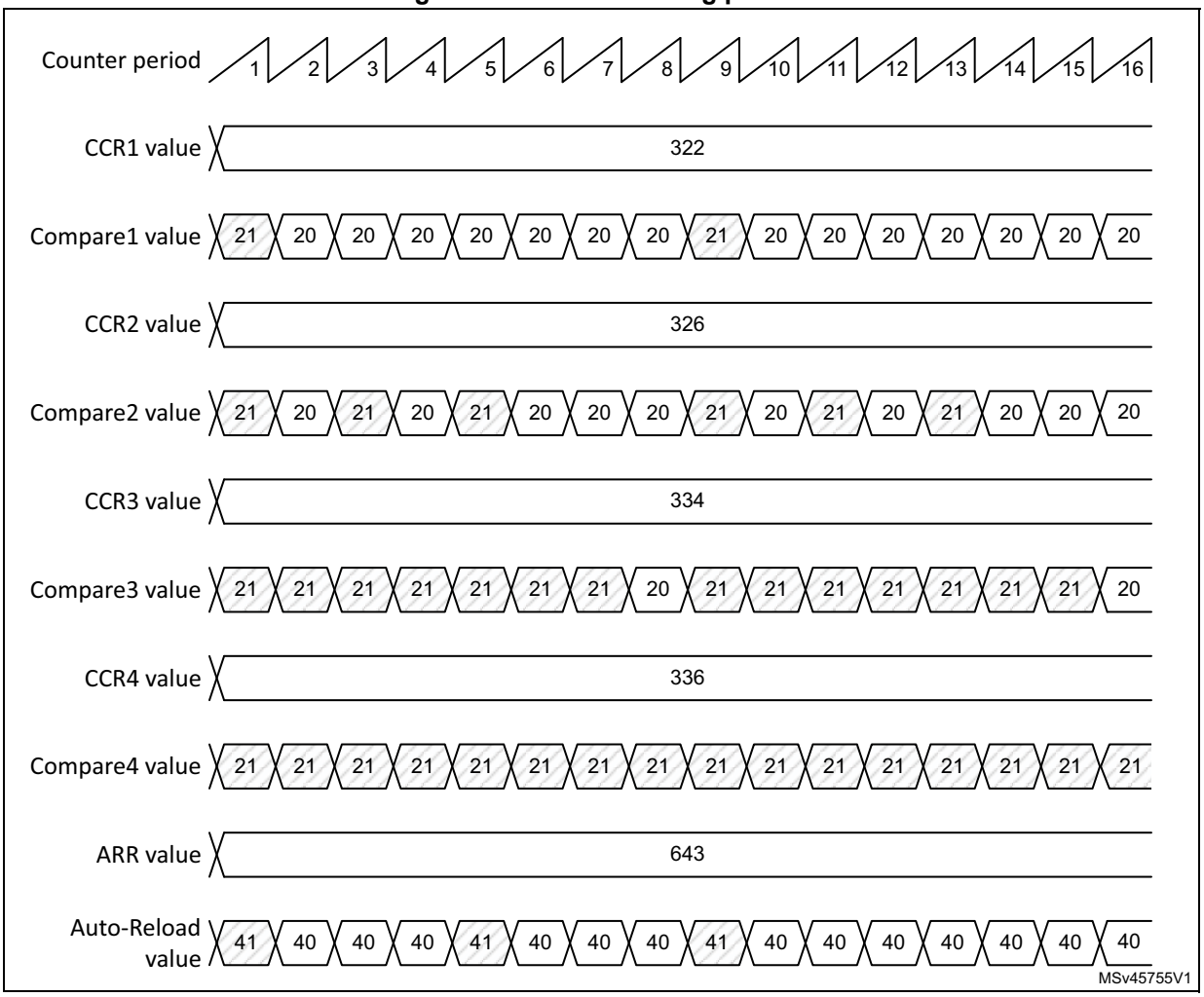


Figure 657. PWM resolution vs frequency (32-bit mode)



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 658*.

Figure 658. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 402*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 402. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIM_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down-counting phases as shown in *Figure 659*.

Figure 659. Dithering effect on duty cycle in center-aligned PWM mode

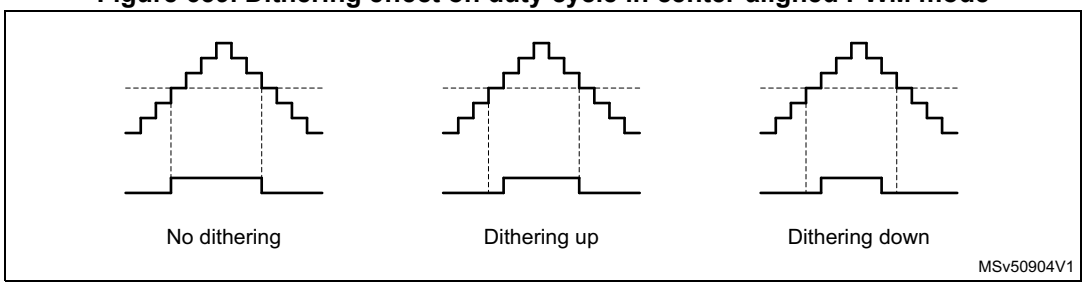


Table 403 shows how the dithering pattern is added in center-aligned PWM mode.

Table 403. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

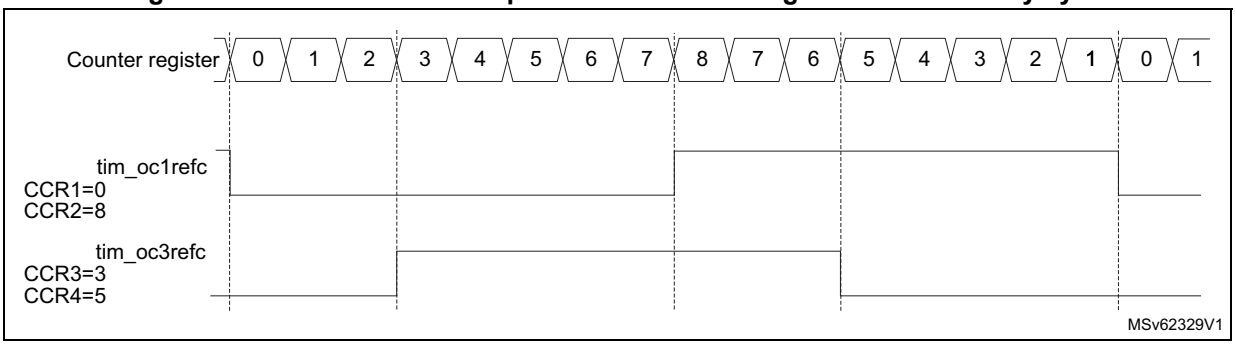
35.4.13 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and the phase-shift are determined by a pair of TIM_CCRx registers. One register controls the PWM during up-counting, the second during down-counting, so that PWM is adjusted every half PWM cycle, for instance:

- tim_oc1refc is controlled by TIM_CCR1 and TIM_CCR2.
- tim_oc3refc is controlled by TIM_CCR3 and TIM_CCR4.

Figure 660 shows an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2).

Figure 660. Generation of two phase-shifted PWM signals with 50% duty cycle



Multiple asymmetric PWM modes are available to combine different pairs of TIM_CCRx registers and generate the required waveform, as listed in *Table 404*. The selection is done with the OCxM[4:0] bitfield in TIM_CCMRx registers. For each pair of TIM_CCRx registers, two options are given (e.g. Asymmetric PWM mode 1 and 2), one being in PWM mode 1 (channel active as long as the counter is below the TIM_CCRx value), the other one in PWM mode 2 (channel active when the counter is above the TIM_CCRx value).

For the asymmetric PWM modes 1 and 2, it is mandatory to have the same PWM mode for both channels. For instance, it is mandatory to have OC1M[4:0] = OC2M[4:0] = 01110 for the asymmetric PWM mode 1 on oc1refc.

For the asymmetric PWM modes 7 to 10, it is mandatory to have opposite PWM mode for both channels (one being in PWM mode 1 and the other in PWM mode 2). Refer to *Table 404* for configuration.

Note: The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bits are not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the regular tim_oc2ref signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 2.

Table 404. Asymmetric PWM modes(1)

Output compare mode	OCxM[4:0]	oc1refc	oc2refc	oc3refc	oc4refc	PWM Mode(2)
Asymmetric PWM mode 1	01110	oc1ref (up-counting) oc2ref (down-counting)		oc3ref (up-counting) oc4ref (down-counting)		1
Asymmetric PWM mode 2	01111					2
Asymmetric PWM mode 7	10110	-	-	oc3refc = 1 (up-counting) oc3ref OR oc4ref (down-counting)	-	1(3)
Asymmetric PWM mode 8	10111	-	-	oc3refc = 0 (up-counting) oc3ref AND oc4ref (down-counting)	-	2(4)
Asymmetric PWM mode 9	11000	-	-	oc3ref OR oc4ref (up-counting) oc3refc = 1 (down-counting)	-	1(3)
Asymmetric PWM mode 10	11001	-	-	oc3ref AND oc4ref (up-counting) oc3refc = 0 (down-counting)	-	2(4)

1. Asymmetric PWM mode 7 to 10 are only available on OC3M.

2. PWM mode 1: channel active as long as the counter is below the TIM_CCRx value. PWM mode 2: channel active when the counter is above the TIM_CCRx value. Unless explicitly mentioned, the PWM mode must be identical for the two channels used for the asymmetric mode.
3. oc4ref must be programmed in PWM mode 1.
4. oc4ref must be programmed in PWM mode 2.

35.4.14 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and delay are determined by the two TIM_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs,

- tim_oc1refc (or tim_oc2refc) is controlled by TIM_CCR1 and TIM_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIM_CCR3 and TIM_CCR4

Table 405 summarizes different combinations of PWM combined mode:

Table 405. Combined PWM mode overview

Output compare mode	OCxM[4:0]	oc1refc	oc2refc	oc3refc	oc4refc	PWM mode(1)
Combined PWM mode 1	01100	oc1ref OR oc2ref		oc3ref OR oc4ref		1
Combined PWM mode 2	01101	oc1ref AND oc2ref		oc3ref AND oc4ref		2

1. PWM mode 1: channel active as long as the counter is below the TIM_CCRx value). PWM mode 2: channel active when the counter is above the TIM_CCRx value.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 01100 (combined PWM mode 1) or 01101 (combined PWM mode 2) in the OCxM bits in the TIM_CCMRx register.

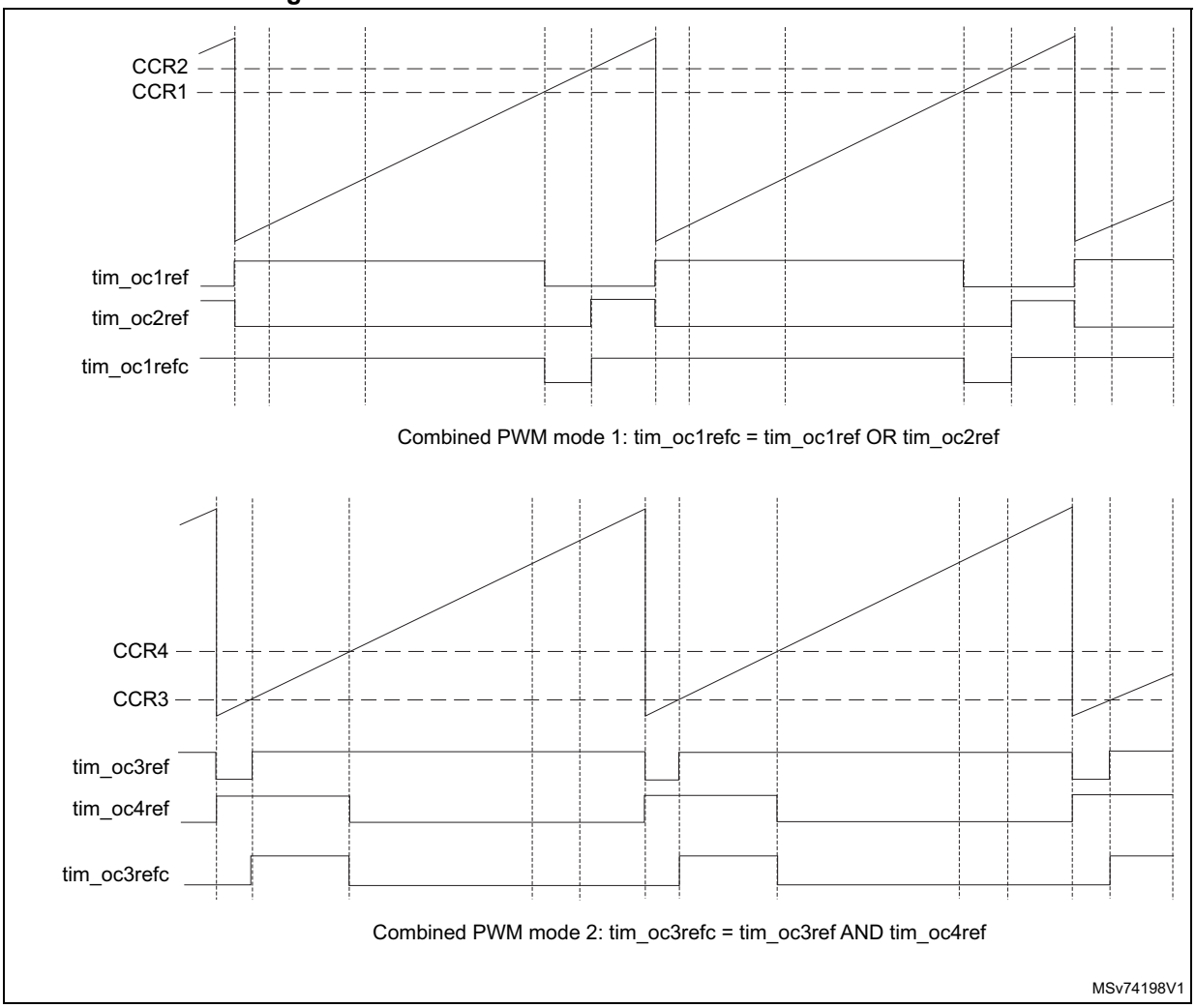
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in PWM mode 1 and the other in PWM mode 2).

Note: The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bits are not contiguous with the three least significant ones.

Figure 661 shows an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in combined PWM mode 1.
- Channel 2 is configured in PWM mode 2.
- Channel 3 is configured in combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.

Figure 661. Combined PWM mode on channels 1 and 3

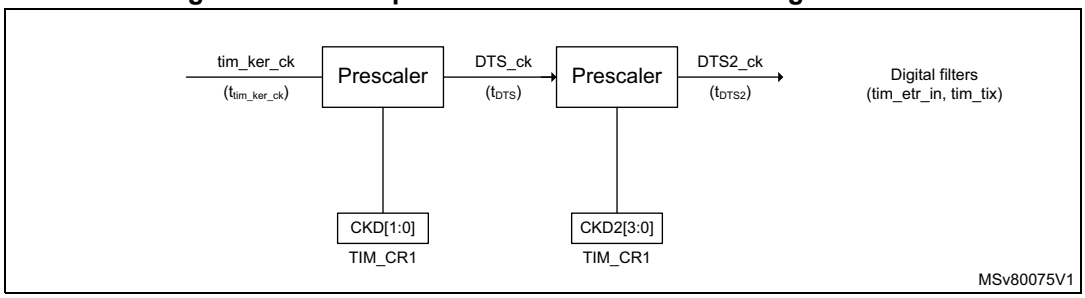


35.4.15 Clock dividers for dead-time generators and digital filters

The digital filters (for tim_etr_in, tim_tix) use a DTS_ck clock (with a t_{DTS} period), derived from the tim_ker_ck clock.

A prescaler can decrease the frequency to extend the DTS_ck clock period programmed with the CKD[1:0] bitfield in the TIM_CR1 register. A second prescaler can decrease the resulting DTS_ck clock to extend the digital filters range, with the CKD2[3:0] bitfield in the TIM_CR1 register.

Figure 662. Clock prescalers for dead-time and digital filters



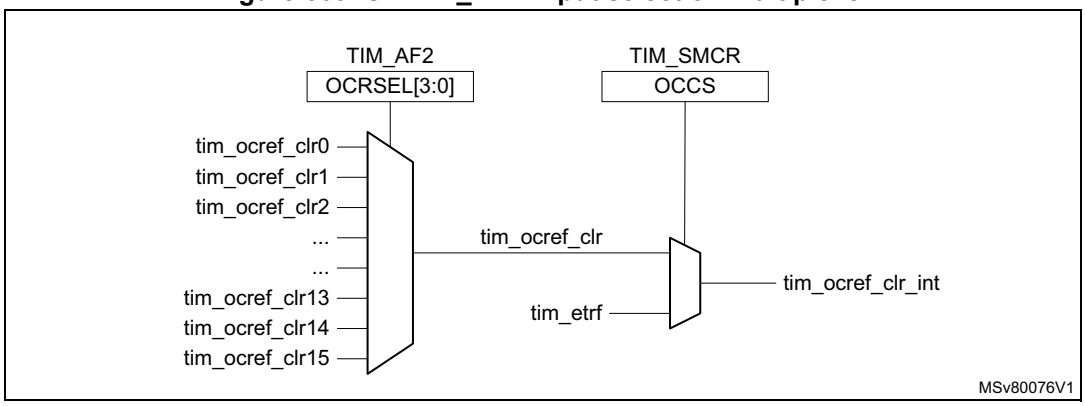
35.4.16 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIM_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in output compare and PWM modes. It must not be used in forced mode.

The tim_ocref_clr_int source depends on the OCREF clear selection feature implementation, refer to *Section 35.3: TIM2/TIM3/TIM4/TIM5 implementation*.

If the OCREF clear selection feature is implemented, the tim_ocref_clr_int can be selected between the tim_ocref_clr input and the tim_etrf input (tim_etr_in after the filter) by configuring the OCCS bit in the TIM_SMCR register. The tim_ocref_clr input can be selected among several tim_ocref_clr[15:0] inputs, using the OCRSEL[3:0] bitfield in the TIM_AF2 register, as shown in *Figure 663*.

Figure 663. OCREF_CLR input selection multiplexer



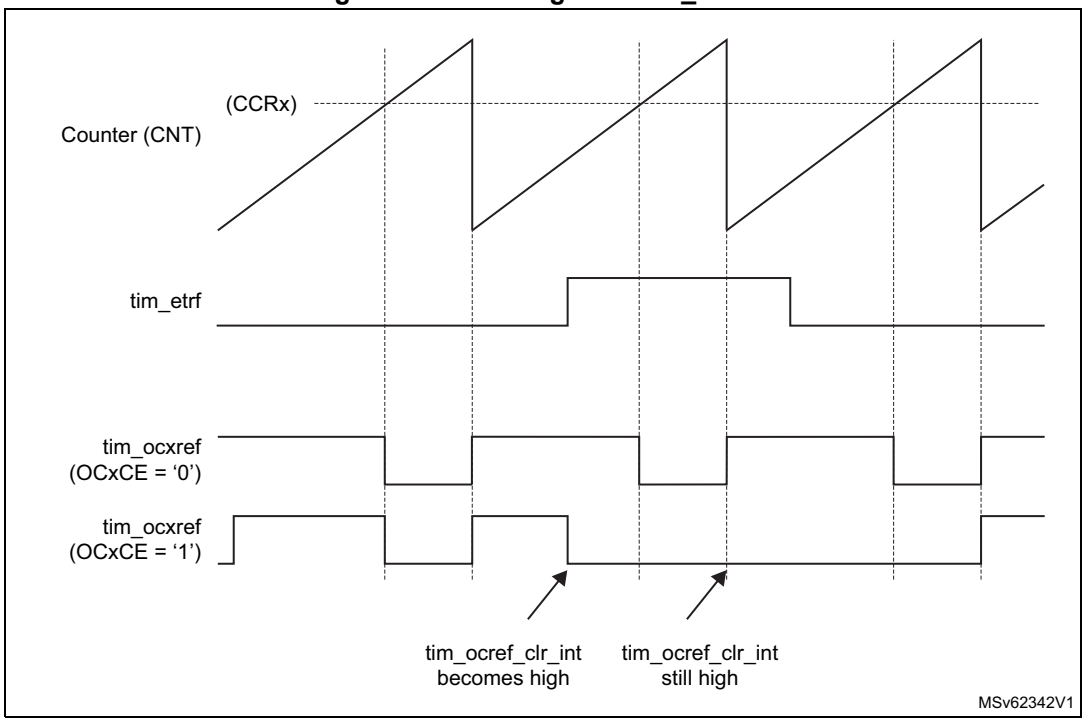
If the OCREF clear selection feature is not implemented, the tim_ocref_clr_int input is directly connected to the tim_etrf input.

For example, the tim_ocref_clr_int signal can be connected to the output of a comparator to be used for current handling. In this case, tim_etr_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] in the TIM_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 664 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Figure 664. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), tim_ocxref is enabled again at the next counter overflow.

This feature is also supported in combined and asymmetric modes. In this case, it is mandatory to enable the OCxCE bit for the channel. The OCxCE bit of the complementary channel (for exemple: OC2CE = X for oc2ref in *Figure 665*) is ignored. *Figure 665* to *Figure 667* describe the output waveforms for specific use cases when the ocref_clr_int signal is asserted.

Figure 665. Asymmetric PWM mode 1

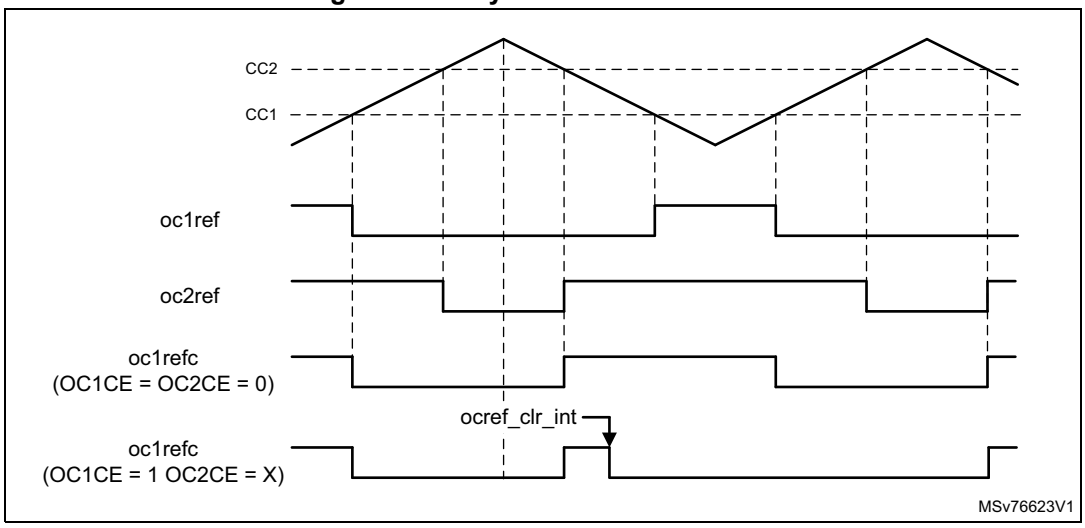


Figure 666. Asymmetric PWM mode 2

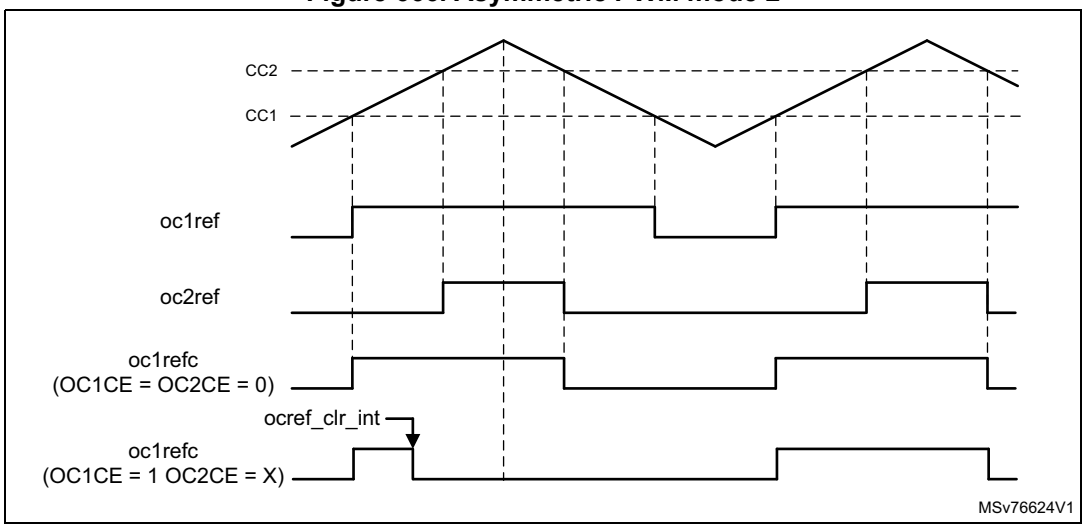
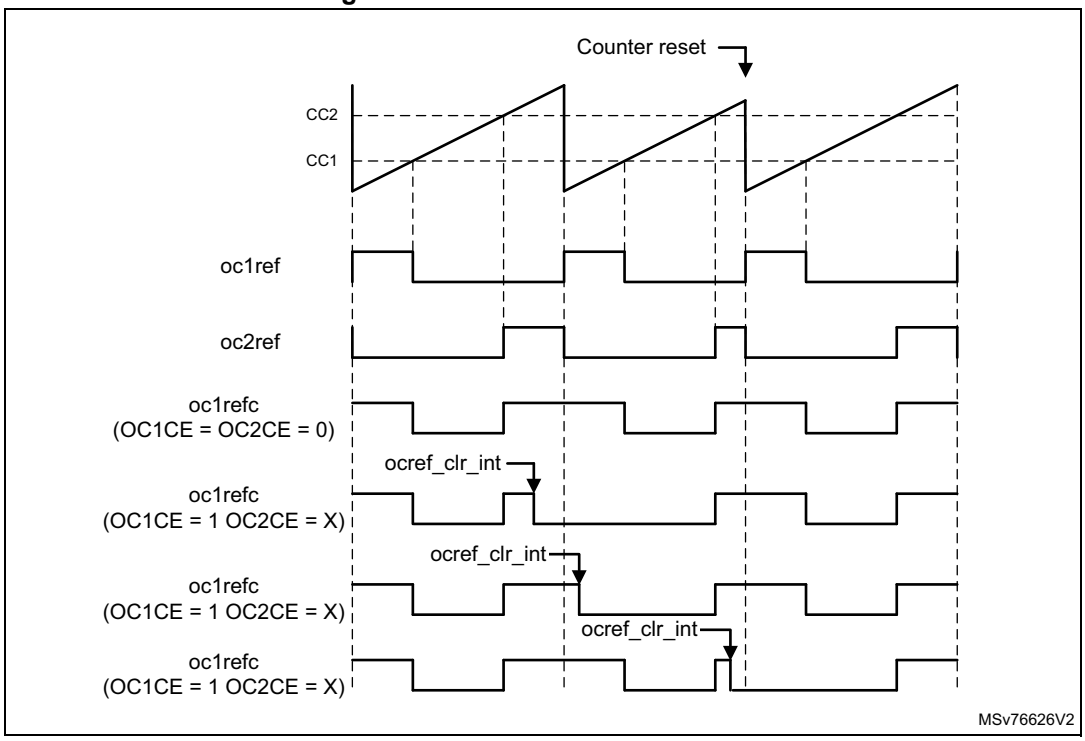


Figure 667. Combined PWM mode 1



Conditions of use

This feature supports cycle-by-cycle current control loops, typically with a comparator shortening the PWM on-time when a current threshold is exceeded.

It cannot be used in the following modes, where the OCxCE bit must be cleared:

- PWM toggle mode (OCxM[4:0] = 0x03)
- One-pulse mode with PWM active on match (OPM = 1 and OCxM[4:0] = 0x06, 0x08, 0x0C, 0x0E)
- Combined PWM mode 1 or 3 in center-aligned mode (CMS[1:0] = 1, 2, 3 and OCxM[4:0] = 0x0C)

35.4.17 One-pulse mode

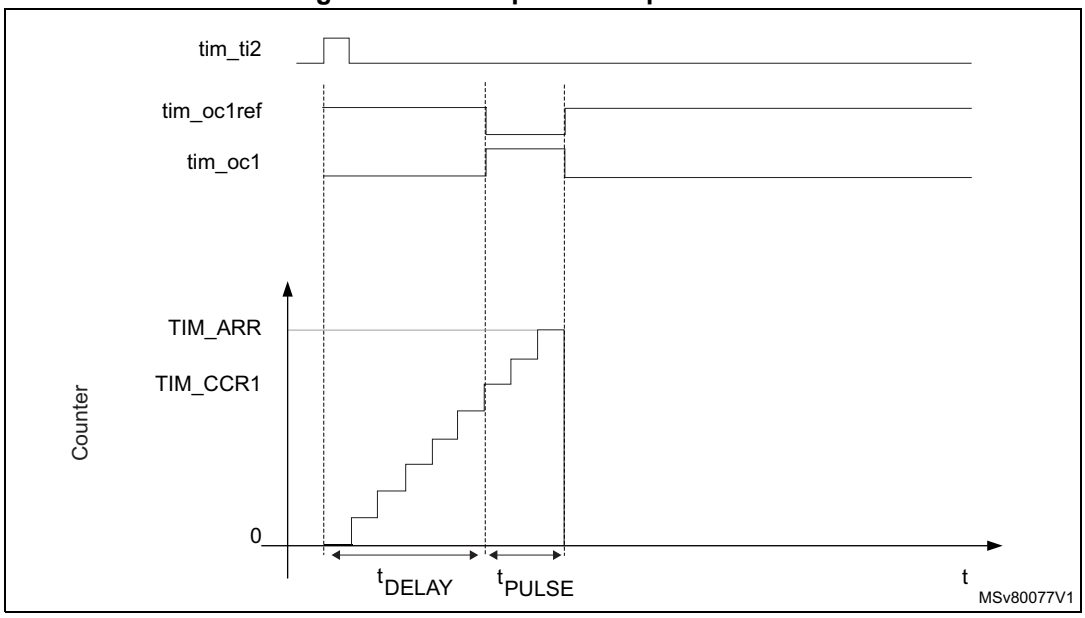
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIM_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

CNT<CCRx ≤ ARR (in particular, 0<CCRx).

Figure 668. Example of one-pulse mode



For example if the user wants to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Map tim_ti2fp2 on tim_ti2 by writing CC2S = 01 in the TIM_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIM_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIM_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIM_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIM_ARR - TIM_CCR1).
- Suppose the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 00111 in the TIM_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIM_CCMR1 register and ARPE in the TIM_CR1 register. In this case one has to write the compare value in the TIM_CCR1 register, the autoreload value in the TIM_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

In this example, the DIR and CMS bits in the TIM_CR1 register must be low.

Since only one pulse (single mode) is needed, a one must be written in the OPM bit in the TIM_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0). When OPM bit in the TIM_CR1 register is set to 0, so the repetitive mode is selected.

Particular case: tim_ocx fast enable:

In one-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIM_CCMRx register. Then tim_ocxref (and tim_ocx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

35.4.18 Retriggerable one-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one-pulse mode described in *Section 35.4.17*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

The timer must be in slave mode, with the bits SMS[4:0] = 01000 (combined reset + trigger mode) in the TIM_SMCR register, and the OCxM[4:0] bits set to 01000 or 01001 for retriggerable OPM mode 1 or 2.

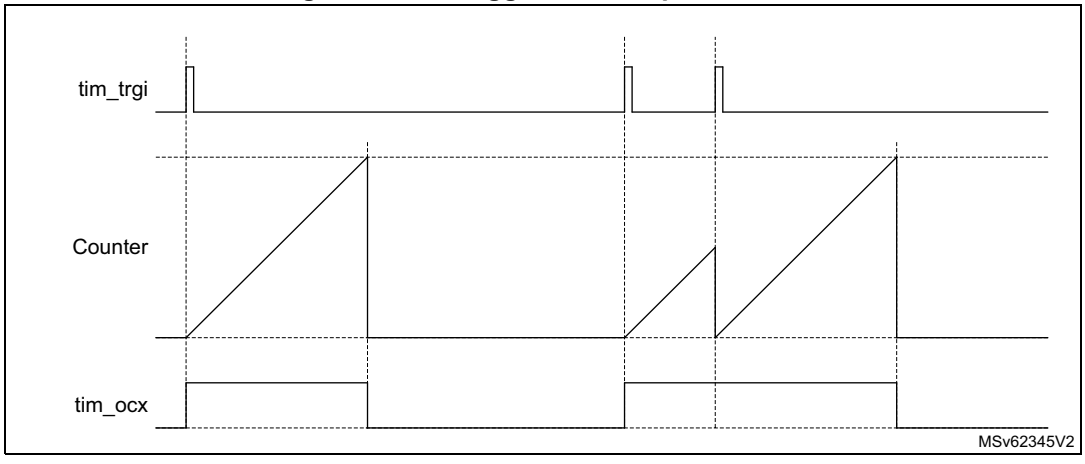
If the timer is configured in up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode CCRx must be above or equal to ARR.

Note: In retriggerable one-pulse mode, the CCxIF flag is not significant.

The OCxM[4:0] and SMS[4:0] bitfields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIM_CR1.

Figure 669. Retriggerable one-pulse mode



35.4.19 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulse width generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on *Figure 670*.

Figure 670. Pulse generator circuitry

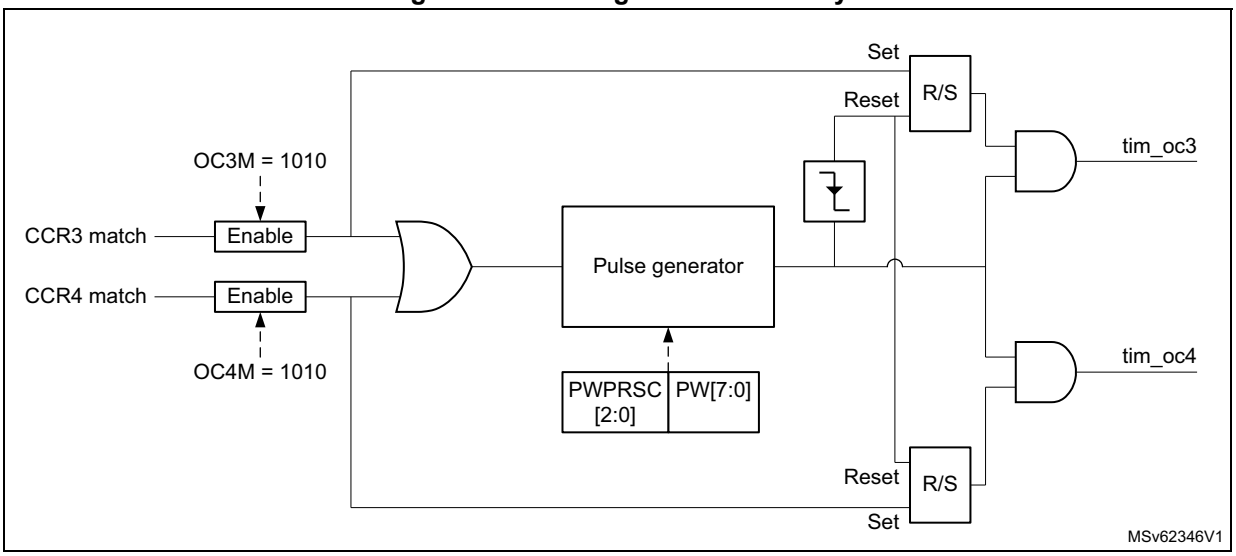
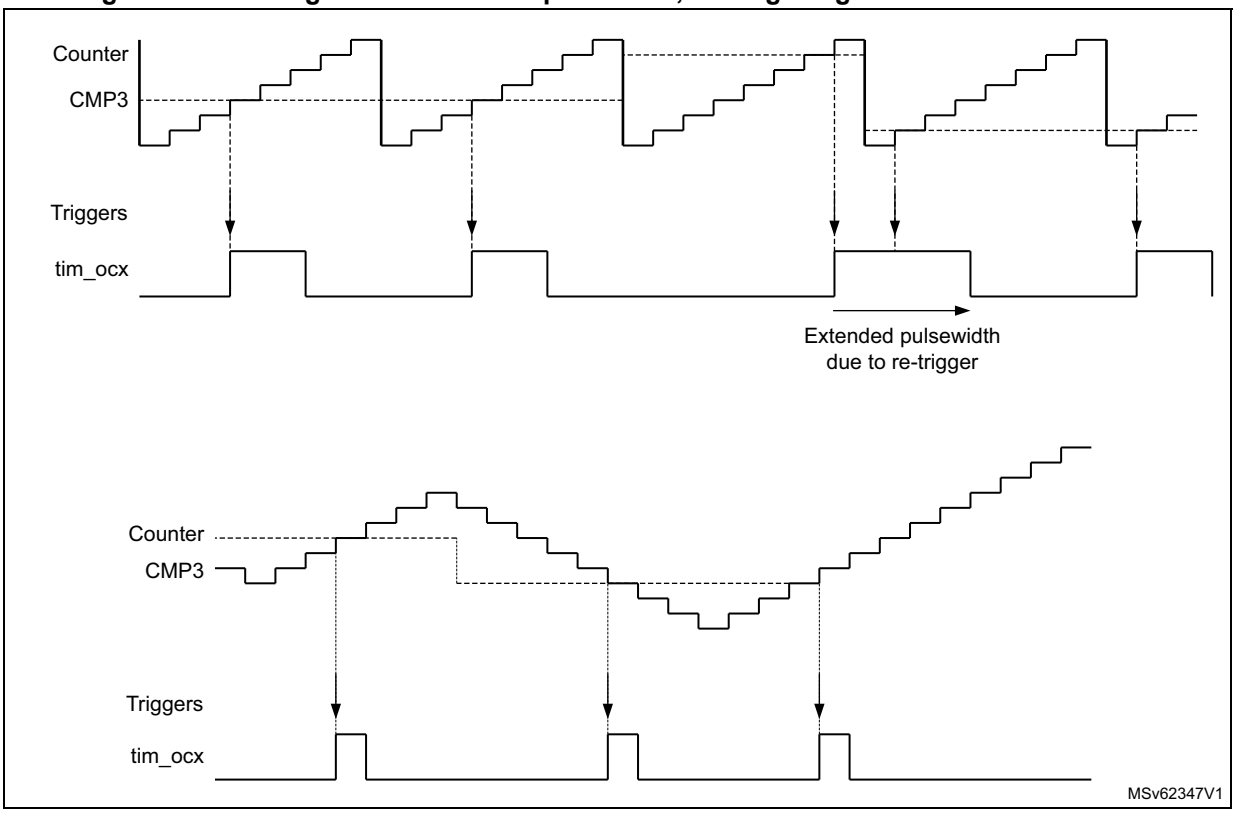


Figure 671 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 671. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bitfields in TIM_CCMR2 register.

The pulse width is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

t_{PW} = PW[7:0] x t_{PWG}

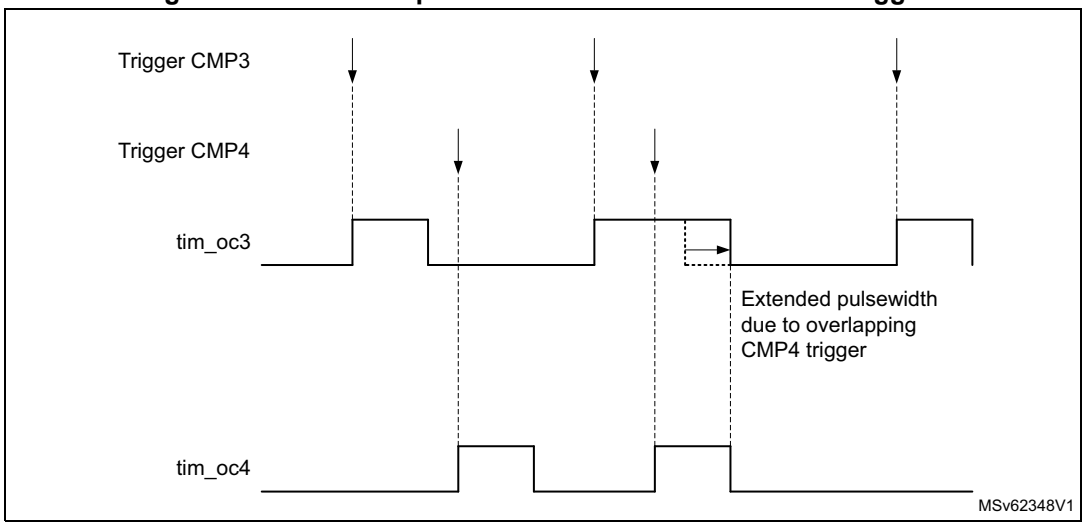
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$.

Gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 672).

Figure 672. Extended pulse width in case of concurrent triggers



35.4.20 Encoder interface mode

Quadrature encoder

To select encoder interface mode write SMS = 0001 in the TIM_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 0010 if it is counting on tim_ti2 edges only and SMS = 0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIM_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well.

The two inputs tim_ti1 and tim_ti2 are used to interface to an incremental encoder. Refer to *Table 406*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIM_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIM_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIM_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIM_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal. Encoder mode and external clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction corresponds to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 406. Counting direction versus encoder signals(CC1P = CC2P = 0)

Active edge	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	01110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	01111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	00001	High	Down	Up	No count	No count
		Low	Up	Down	No count	Down
Counting on tim_ti2 only x2 mode	00010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	00011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicates the mechanical zero position, can be connected to the external trigger input and trigger a counter reset.

Figure 673 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S = 01 (TIM_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIM_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P and CC1NP = 0 (TIM_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P and CC2NP = 0 (TIM_CCER register, tim_ti2fp2 noninverted, tim_ti2fp2 = tim_ti2).
- SMS = 0011 (TIM_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIM_CR1 register, counter is enabled).

Figure 673. Example of counter operation in encoder interface mode

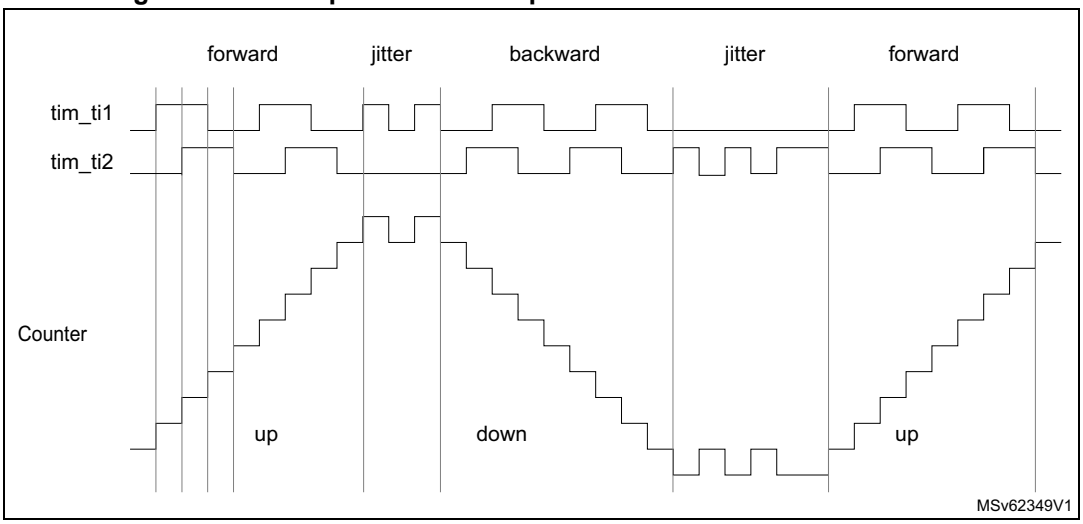


Figure 674 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 674. Example of encoder interface mode with tim_ti1fp1 polarity inverted

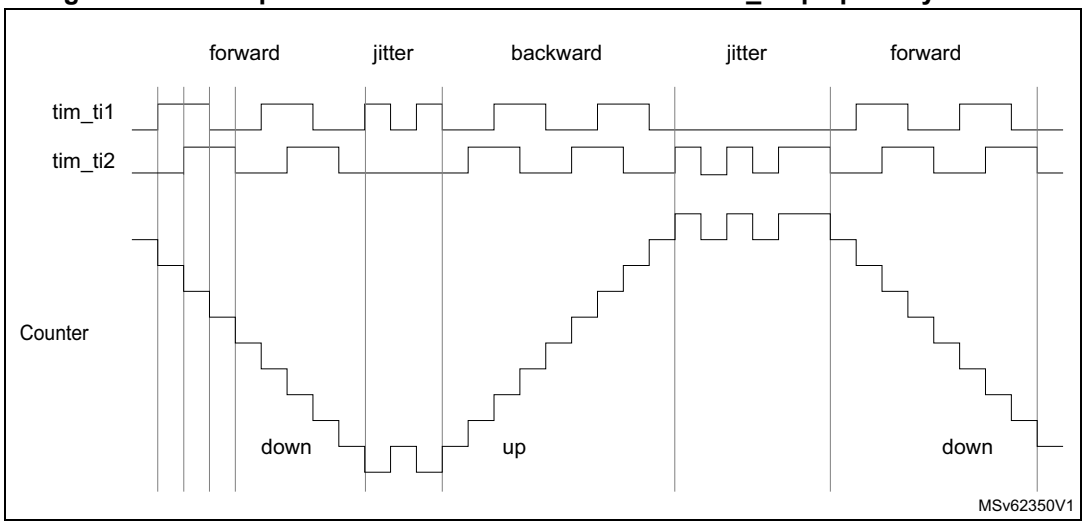
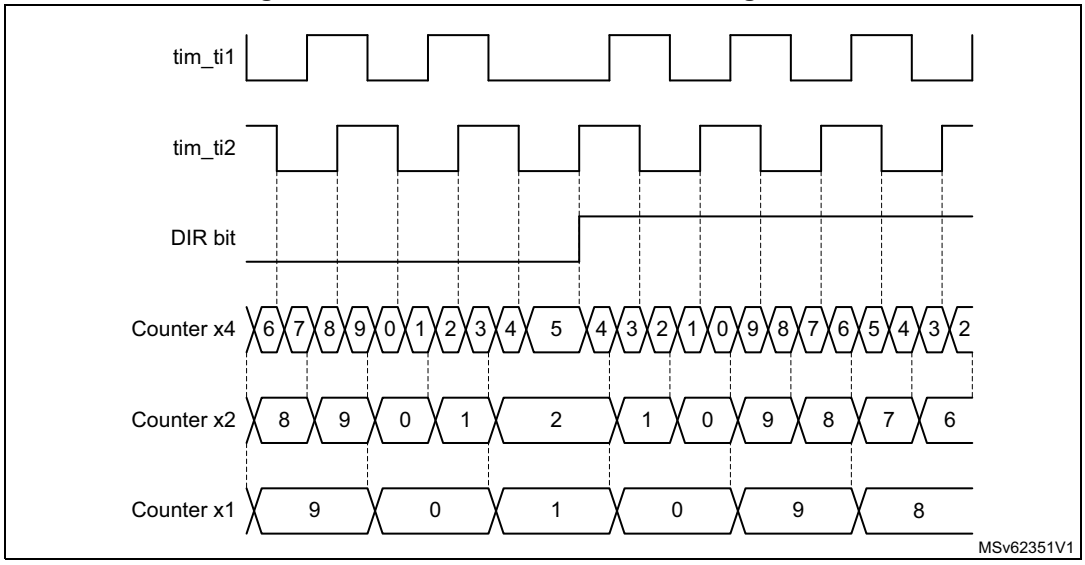


Figure 675 shows the timer counter value during a speed reversal, for various counting modes.

Figure 675. Quadrature encoder counting modes



The timer, when configured in encoder interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIM_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

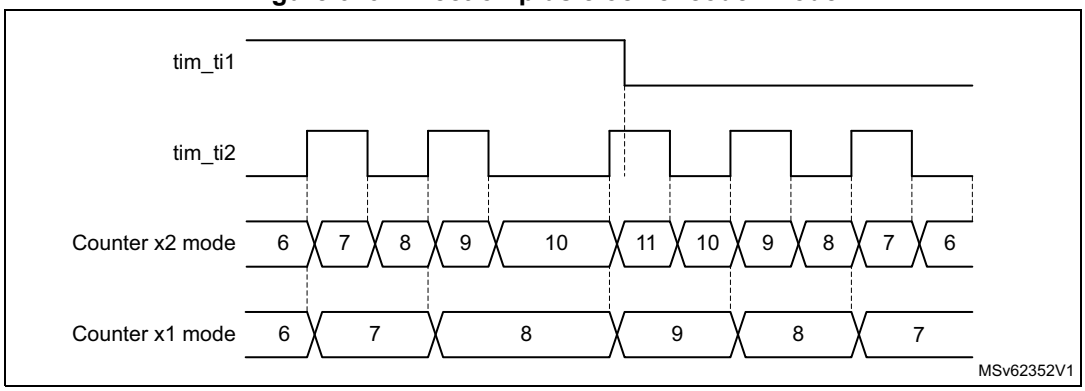
In the "clock plus direction" mode shown on *Figure 676*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01010: x2 mode, the counter is updated on both rising and falling edges of the clock.
- 01011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity.

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 676. Direction plus clock encoder mode



Directional clock encoder mode

In the “directional clock” mode on *Figure 677*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock lines. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 677*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 678*).
- 01101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 677*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 678*).

Figure 677. Directional clock encoder mode (CC1P = CC2P = 0)

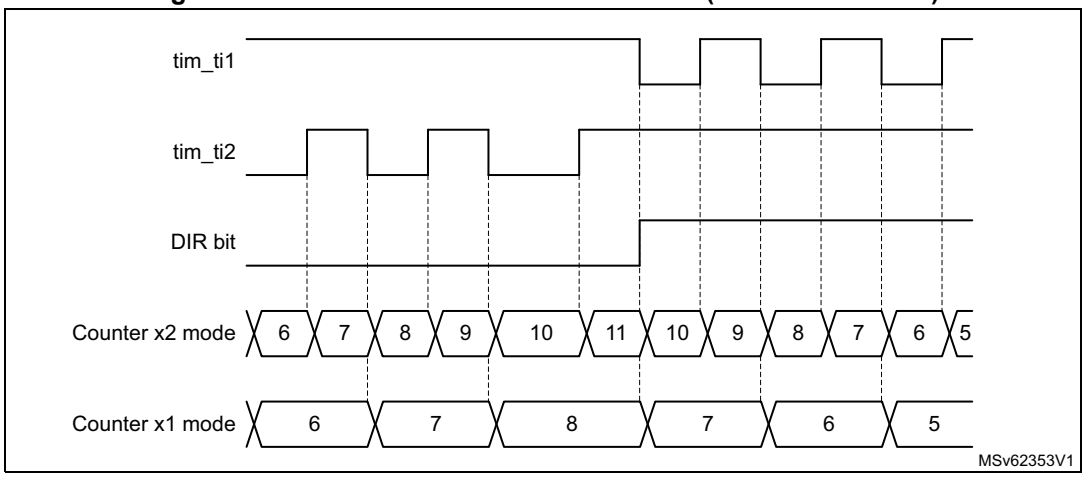


Figure 678. Directional clock encoder mode (CC1P = CC2P = 1)

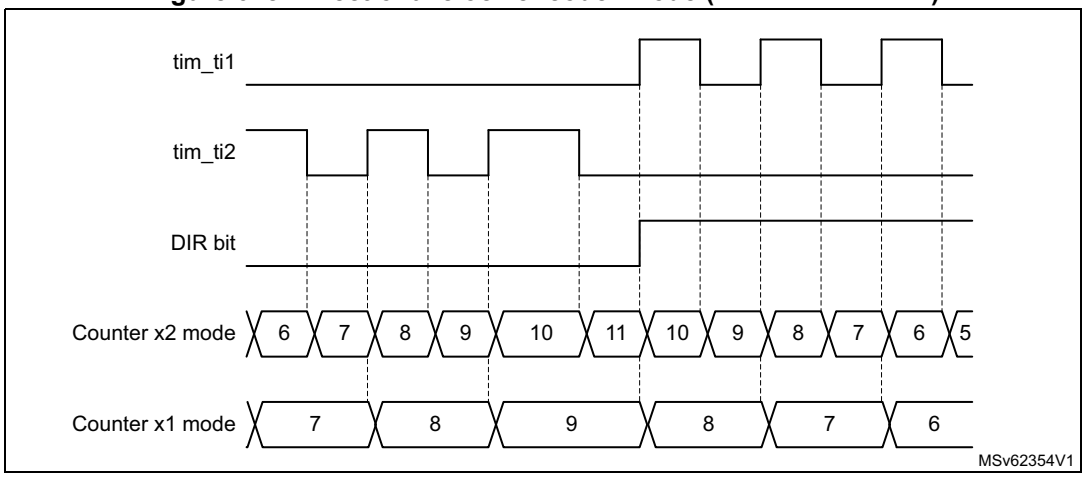


Table 407 details how the directional clock mode operates, for any input transition.

Table 407. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	01100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	01100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	01101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	01101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index input

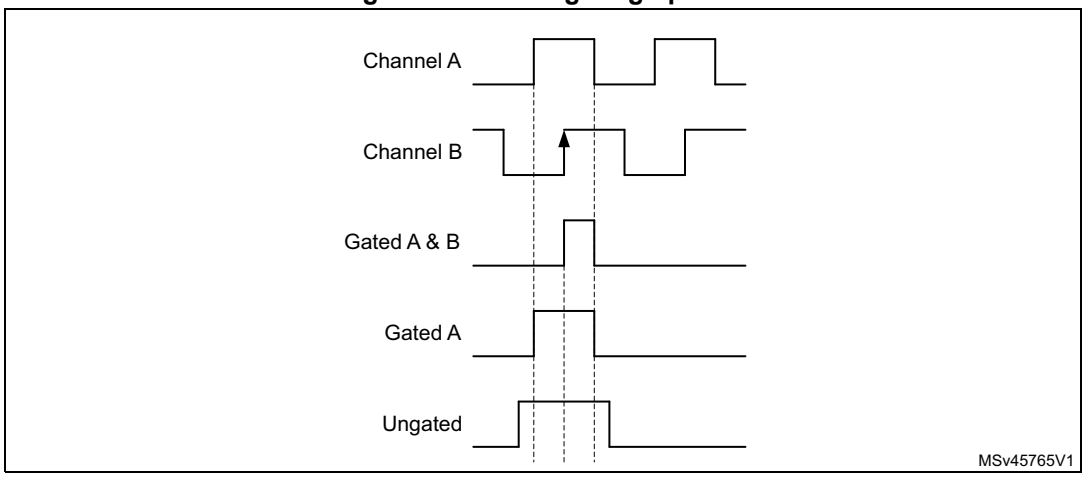
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIM_ECR register. The IE bit must be set only in encoder mode, when the SMS[4:0] bitfield has the following values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111.

Available encoders are proposed with several options for index pulse conditioning, as per *Figure 679*:

- Gated with A and B: the pulse width is 1/4 of one channel period, aligned with both A and B edges.
- Gated with A (or gated with B): the pulse width is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B).
- Ungated: the pulse width is up to one channel period, without any alignment to the edges.

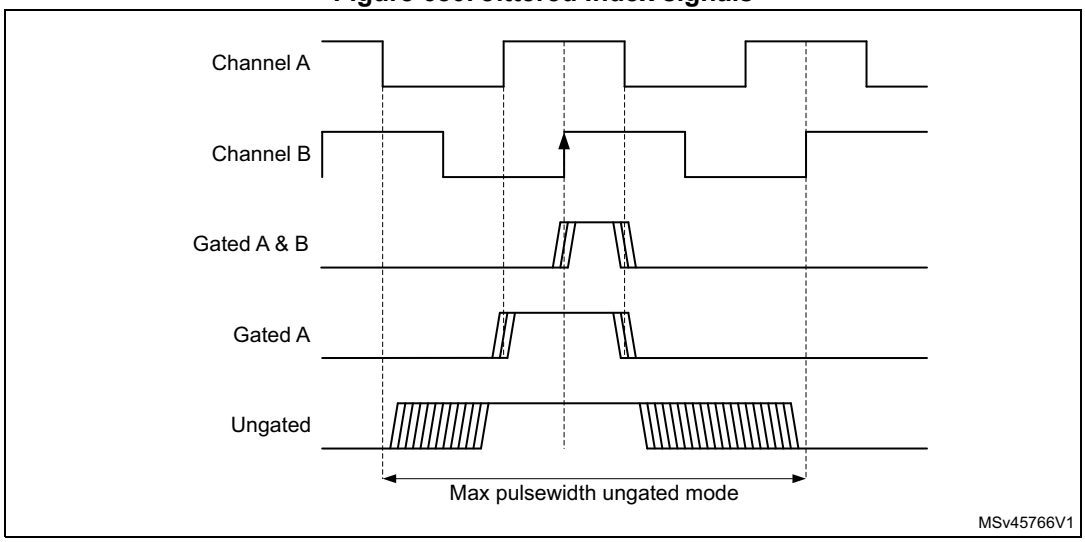
Figure 679. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as shown on *Figure 680*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulse width is greater or equal to two encoder period, the counter is reset multiple times.

Figure 680. Jittered Index signals



The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example channel A and channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIM_ECR register.

The index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIM_ARR when down-counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. *Figure 681* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 681. Index generation for IPOS[1:0] = 11

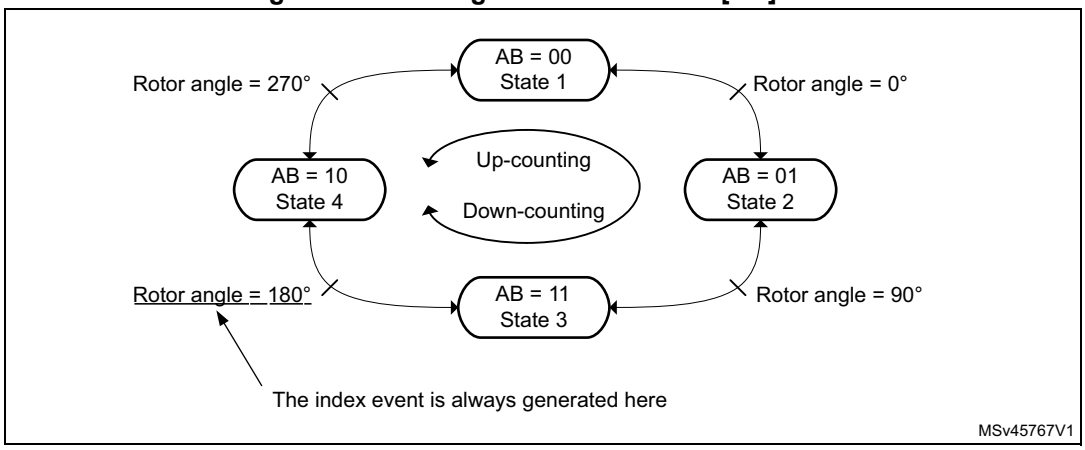


Figure 682 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is 11 (ChA = 1, ChB = 1), when up-counting (DIR bit = 0).
- Counter set to TIM_ARR when exiting the 11 state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

Figure 682. Counter reading with index gated on channel A (IPOS[1:0] = 11)

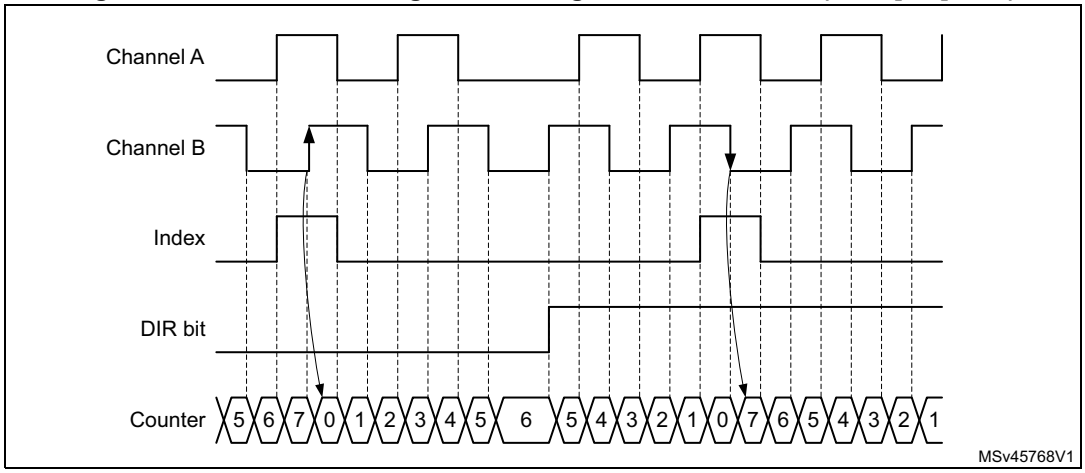


Figure 683 presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 683. Counter reading with index ungated (IPOS[1:0] = 00)

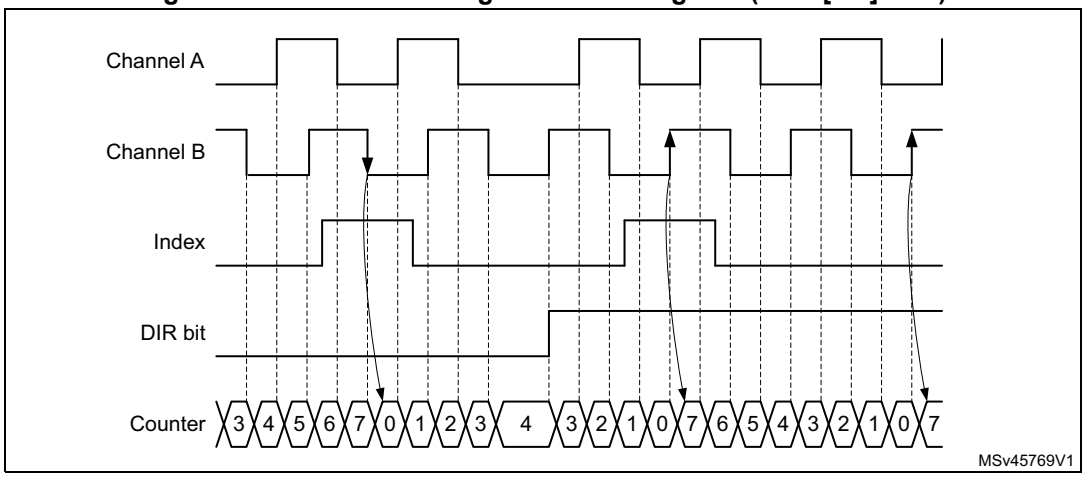


Figure 684 shows how the gated on A & B mode is handled, for various pulse alignment scenarios. The arrows are indicating on which transition is the index event generated.

Figure 684. Counter reading with index gated on channel A and B

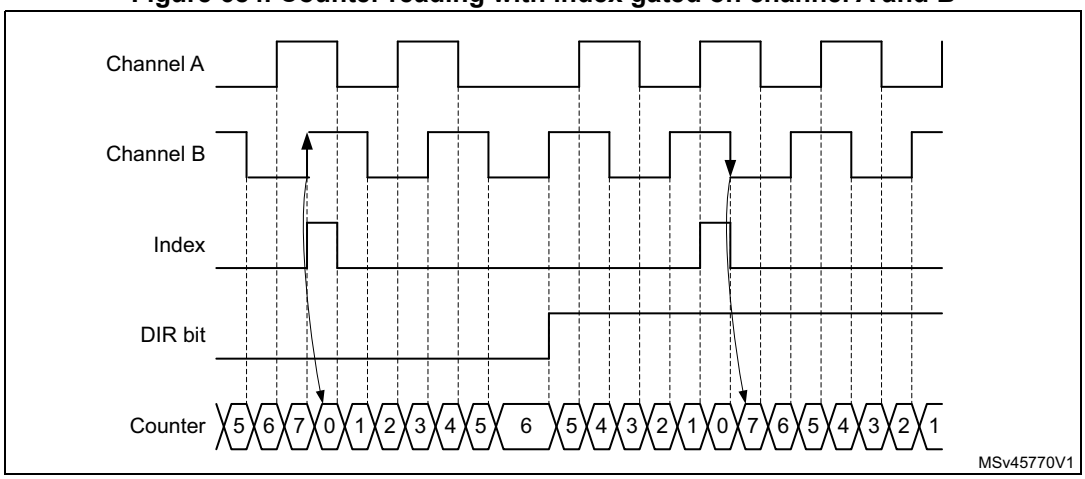


Figure 685 and *Figure 686* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 685. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

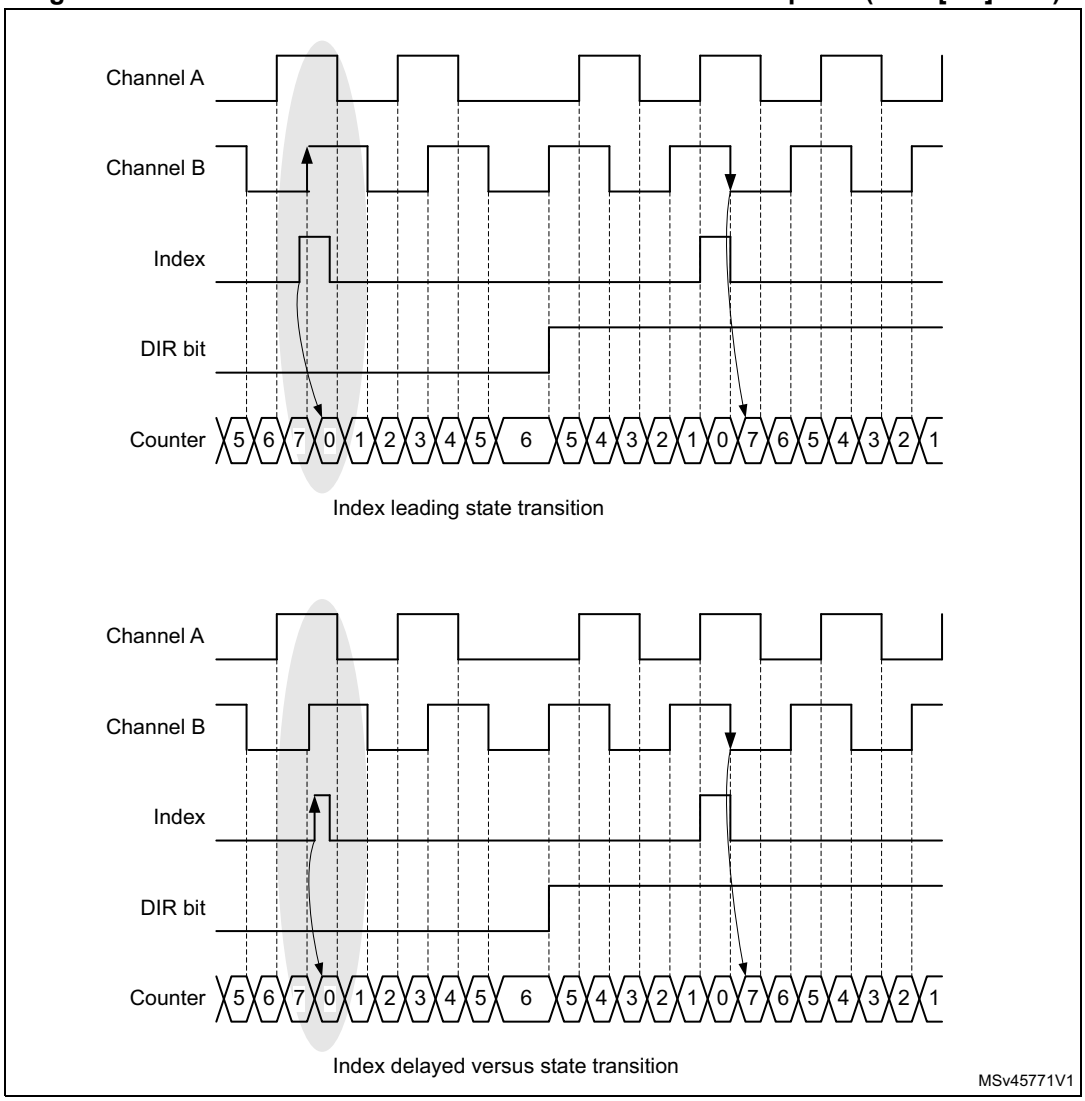


Figure 686. Counter reset Narrow index pulse (closer view, ARR = 0x07)

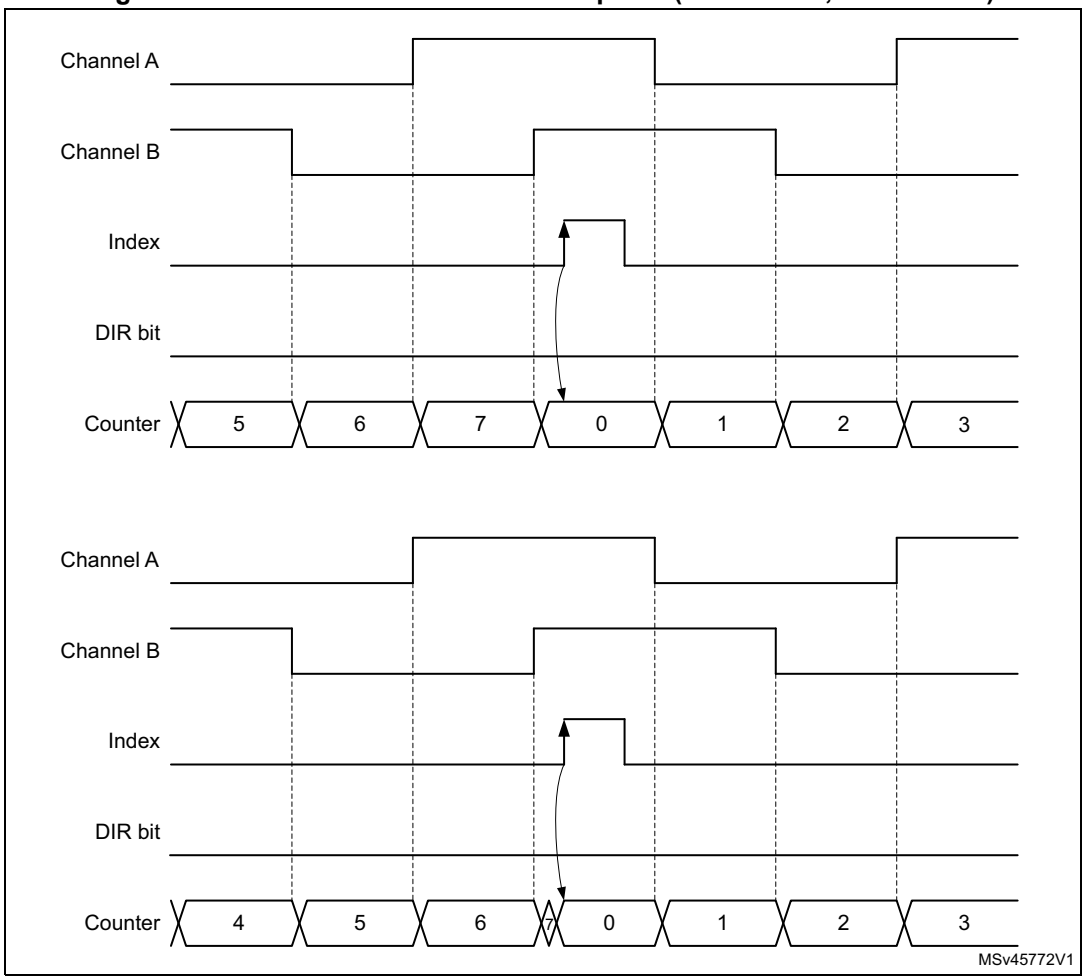
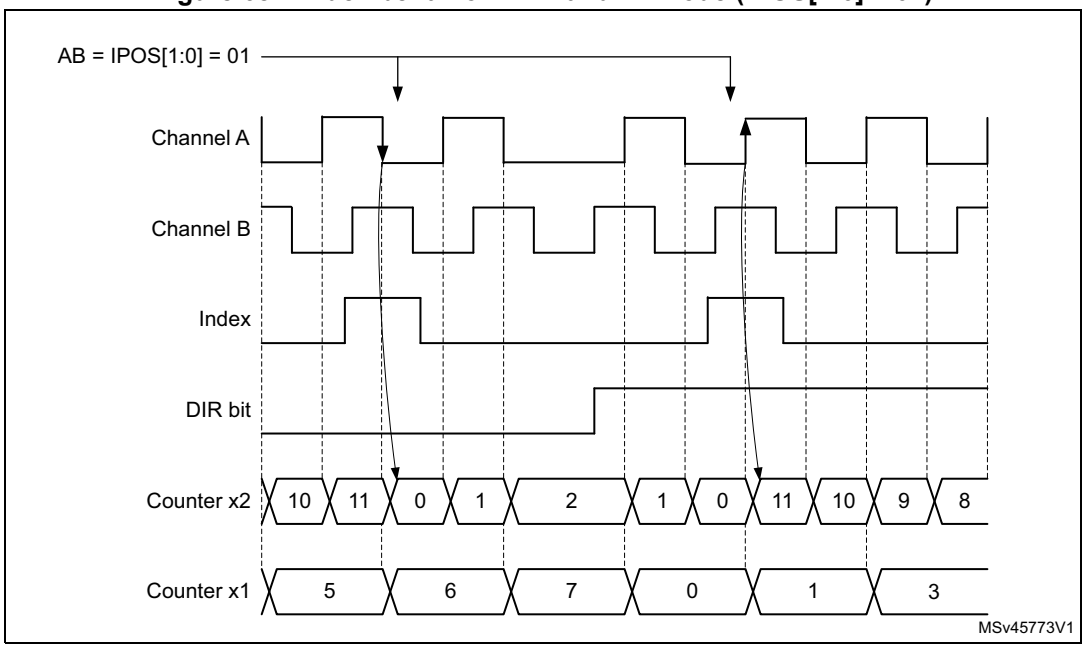


Figure 687 shows how the index is managed in x1 and x2 modes.

Figure 687. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

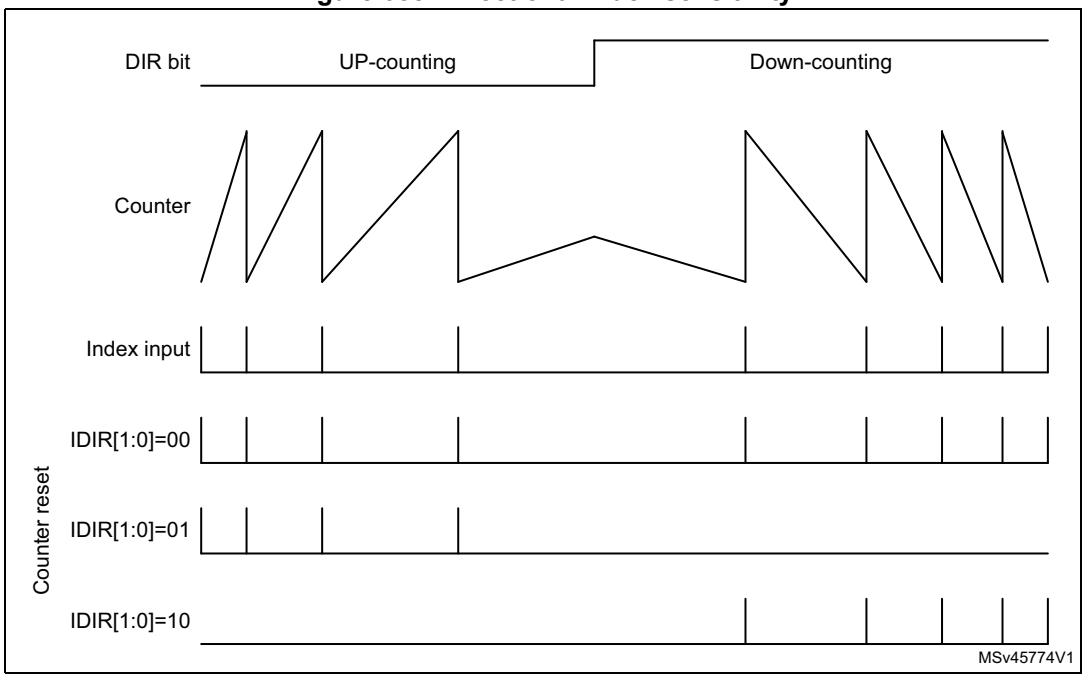


Directional index sensitivity

The IDIR[1:0] bitfield in the TIM_ECR register allows the index to be active only in a selected counting direction.

Figure 688 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

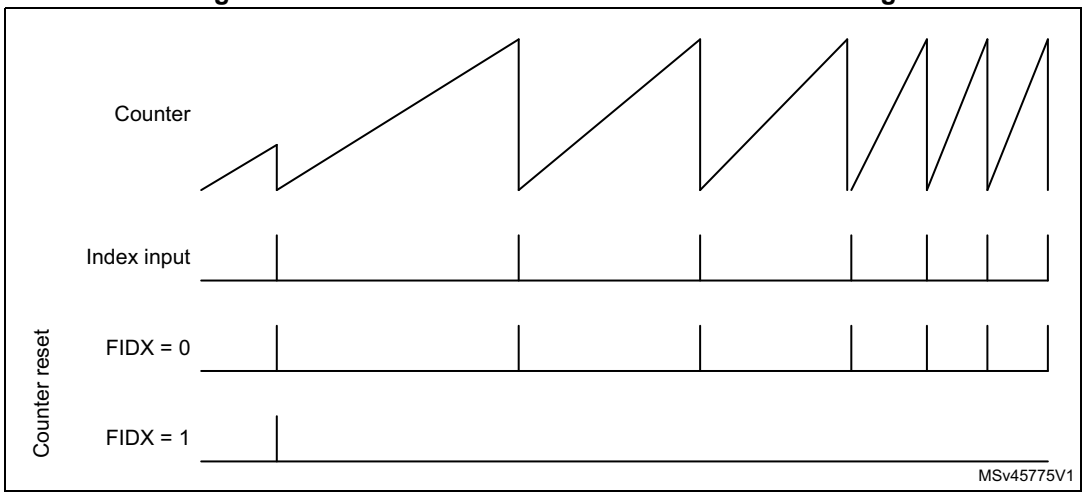
Figure 688. Directional index sensitivity



Special first index event management

The FIDX bit in the TIM_ECR register allows the index to be taken only once, as shown on *Figure 689*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

Figure 689. Counter reset as function of FIDX bit setting



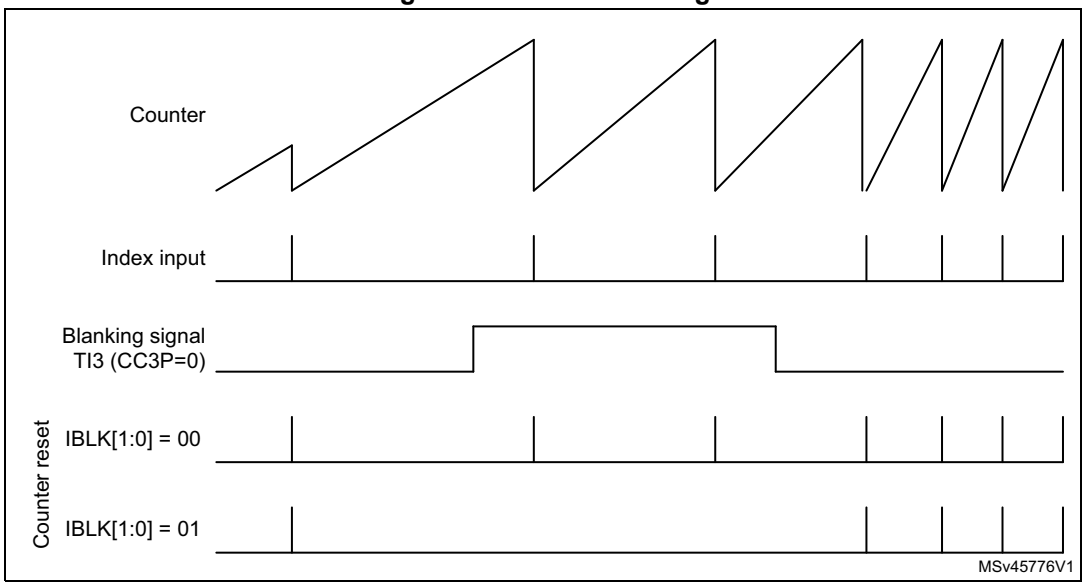
Index blanking

The index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on *Figure 690*.

This mode is enabled using the IBLK[1:0] bitfield in the TIM_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active.
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input.
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input.

Figure 690. Index blanking



Index management in nonquadrature mode

Figure 691 and *Figure 692* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[4:0] bitfield is equal to 01010, 01011, 01100, 01101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level.
- IPOS[0] = 1: Index is detected on clock high level.

The IPOS[1] bit is not-significant.

Figure 691. Index behavior in clock + direction mode, IPOS[0] = 1

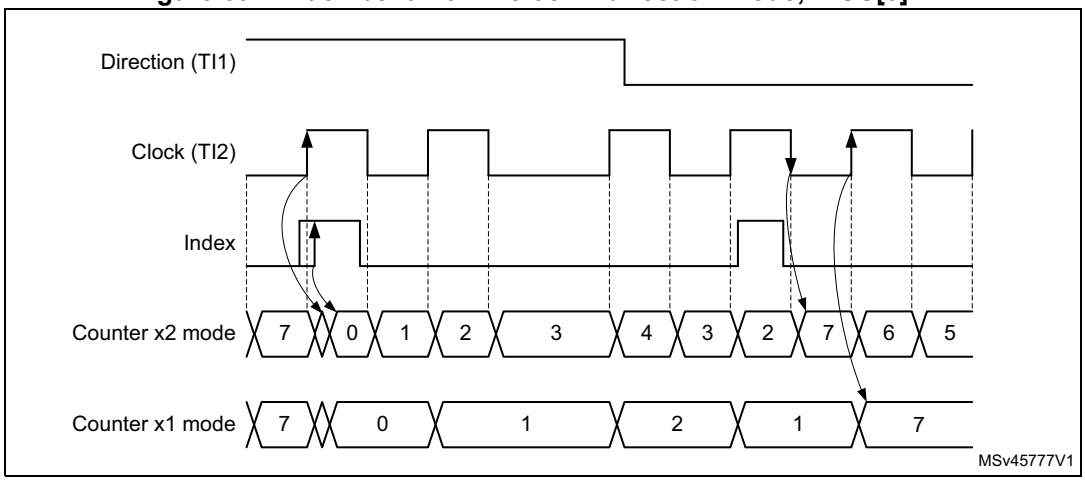
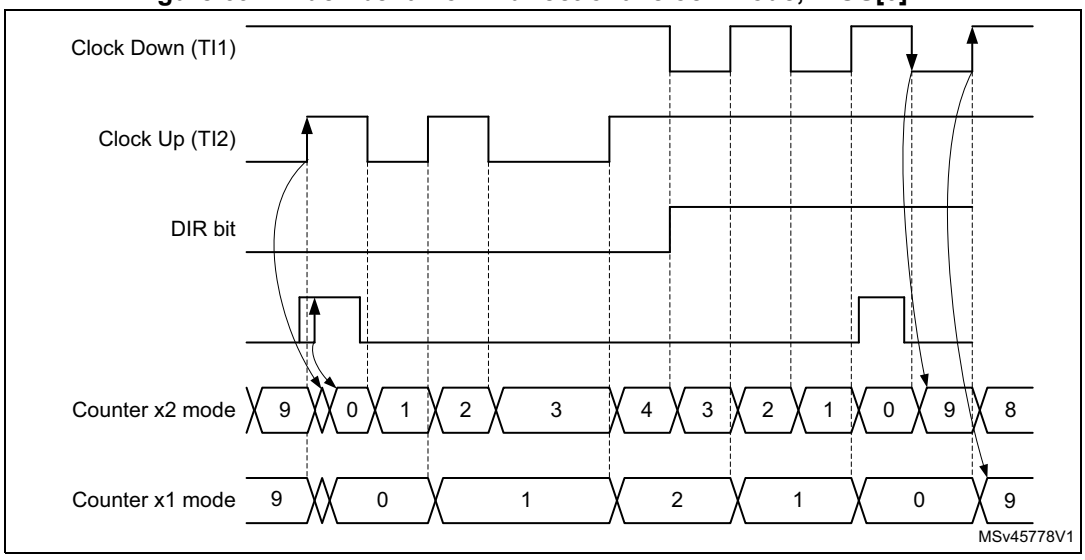


Figure 692. Index behavior in directional clock mode, IPOS[0] = 1

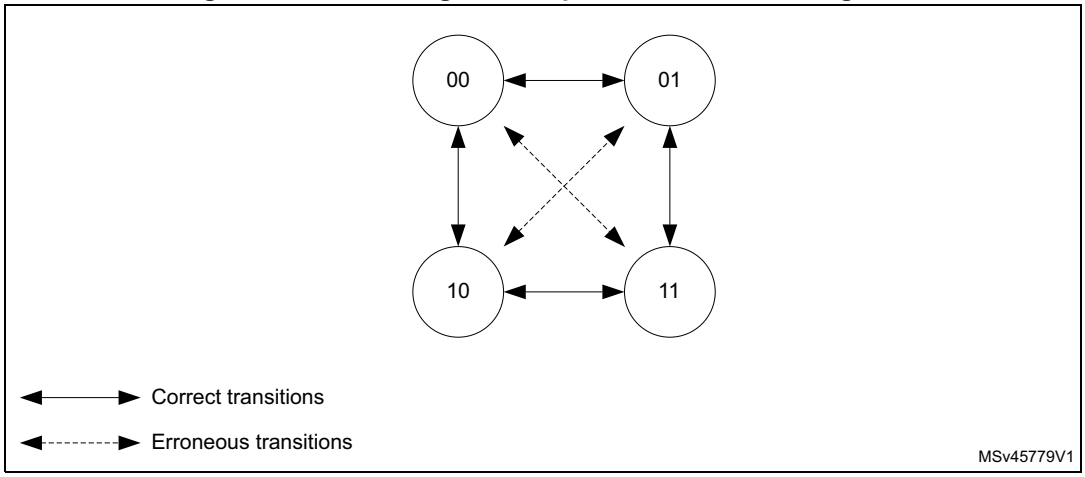


Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 693*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIM_SR status

register. A transition error interrupt is generated if the TERRIE bit is set in the TIM_DIER register.

Figure 693. State diagram for quadrature encoded signals



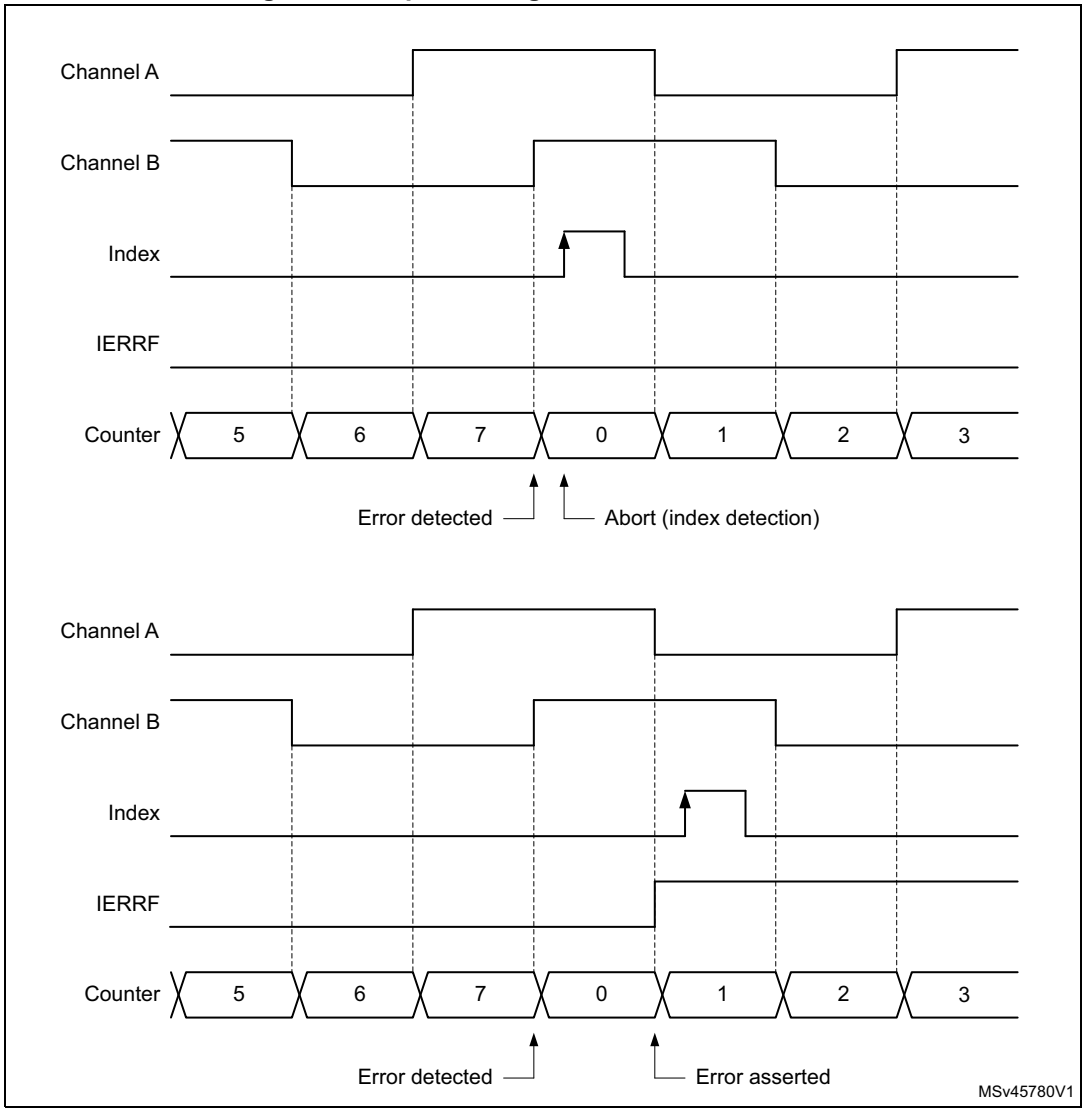
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIM_ARR to 0 or decremented from 0 to TIM_ARR value without any index event, this is reported as an index position error.

The overflow threshold is programmed using the TIM_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIM_ARR = 3999 + 1 = 4000.

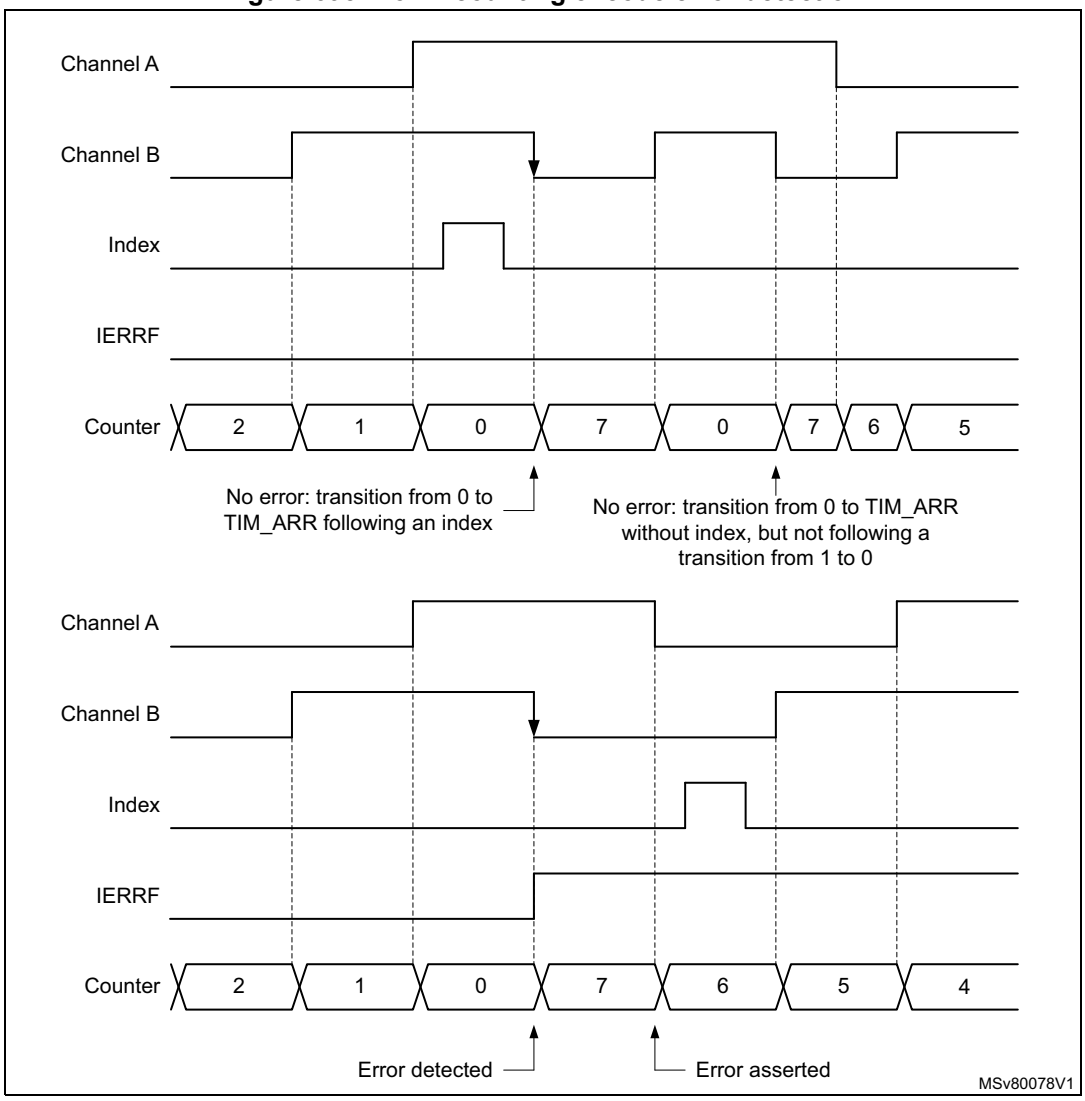
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 694*.

Figure 694. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 695*, to avoid any false error detection in case the encoder dithers between TIM_ARR and 0 immediately after the index detection.

Figure 695. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIM_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIM_DIER register.

Functional encoder interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIM_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIM_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIM_DIER register.
- Index event: the index event sets the IDXF interrupt flag in the TIM_SR status register. An index interrupt is generated if the IDXIE bit is set in the TIM_DIER register.

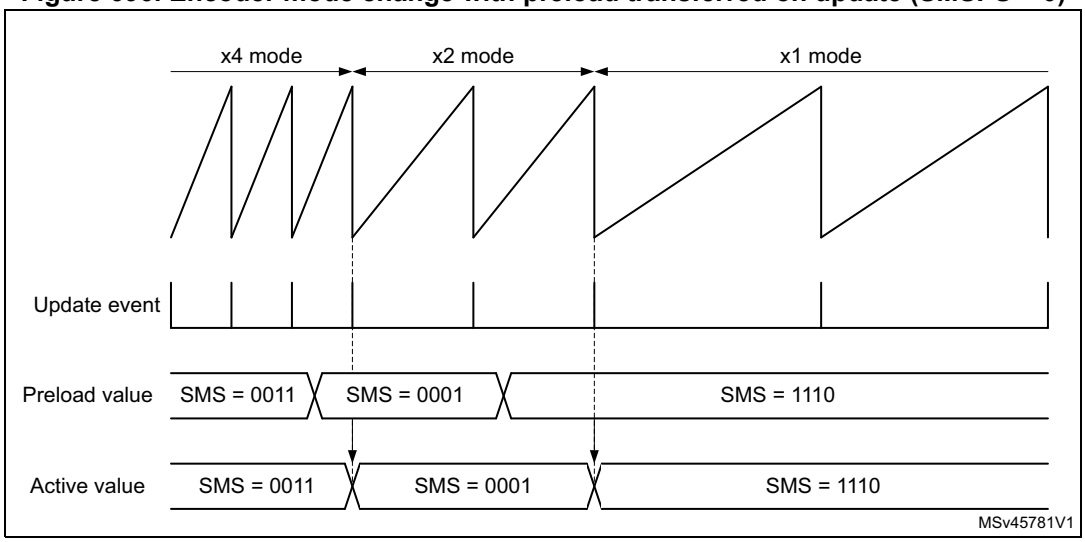
Slave mode selection preload for run-time encoder mode update

It can be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 696*.

For this purpose, the SMS[4:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIM_SMCR register. The trigger for the transfer from SMS[4:0] preload to active value can be selected with the SMSPS bit in the TIM_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when up-counting, and underflows when down-counting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 696. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIM_CR2 register. It is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

35.4.21 Encoder with built-in debouncer

This mode is intended to support noisy digital potentiometers and knobs, providing quadrature encoded signals with potentially long bouncing periods after each transition.

The operating principle consists of working on the first valid transition and discard any other until a second valid transition is detected on the alternate encoder signal.

The *Figure 697* below represents the x2 operating mode. It must be noticed that a transition on input 1 is ignored during direction reversal, if there's no edge on input 2.

The *Figure 698* below represents the x4 operating mode. In this case, a transition is always ignored during a direction reversal or when the encoder dithers between two positions.

These two modes are enabled by writing SMS[4:0] = 10000 or 10001 in TIM_SMCR register.

The encoder with built-in debouncer modes are not suitable for tracking a position or an angle in motor control applications. The transitions ignored during direction reversal will lead to a position/angle error that will build-up over time.

Figure 697. Encoder with built-in debouncer, x2 mode

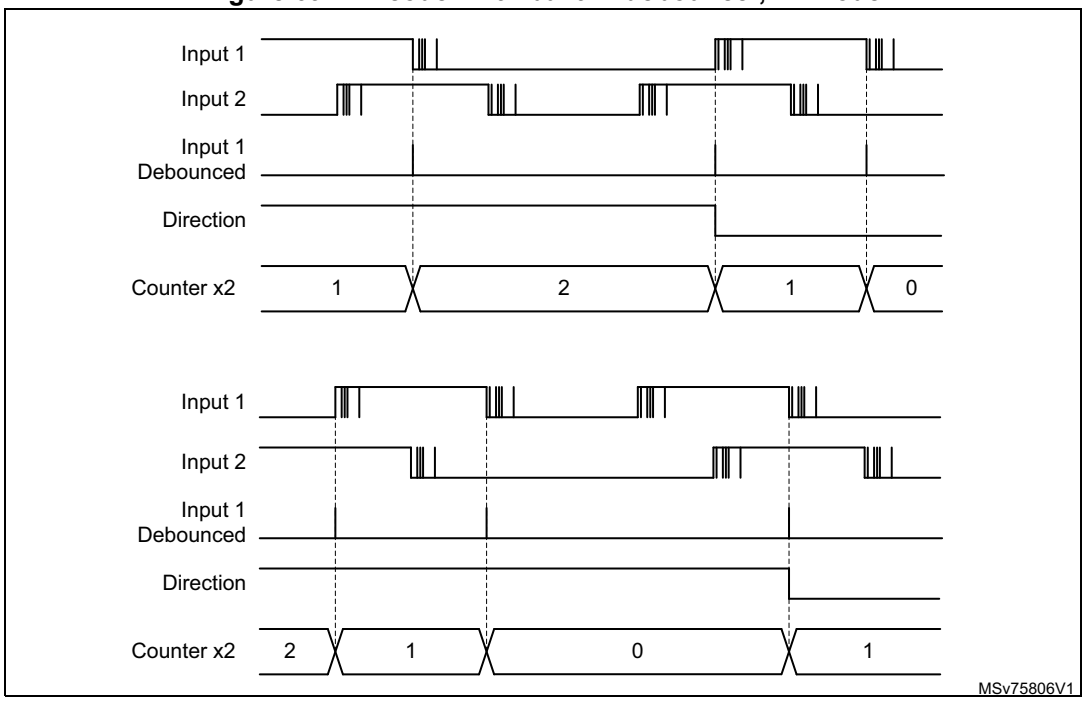
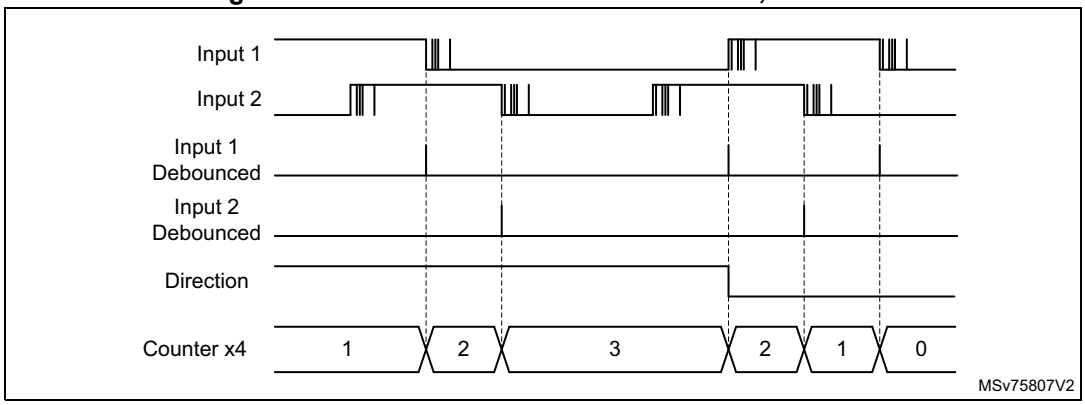


Figure 698. Encoder with built-in debouncer, x4 mode



35.4.22 Direction bit output

It is possible to output a direction signal out of the timer, on the tim_oc3 and tim_oc4 output signals (copy of the DIR bit in the TIM_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bitfield to 1011 in the TIM_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

35.4.23 Clock drift measurement

It is possible to measure the drift between two clock sources using the directional clock encoder mode. This is useful for audio applications typically, when one need to compute the drift between two streams.

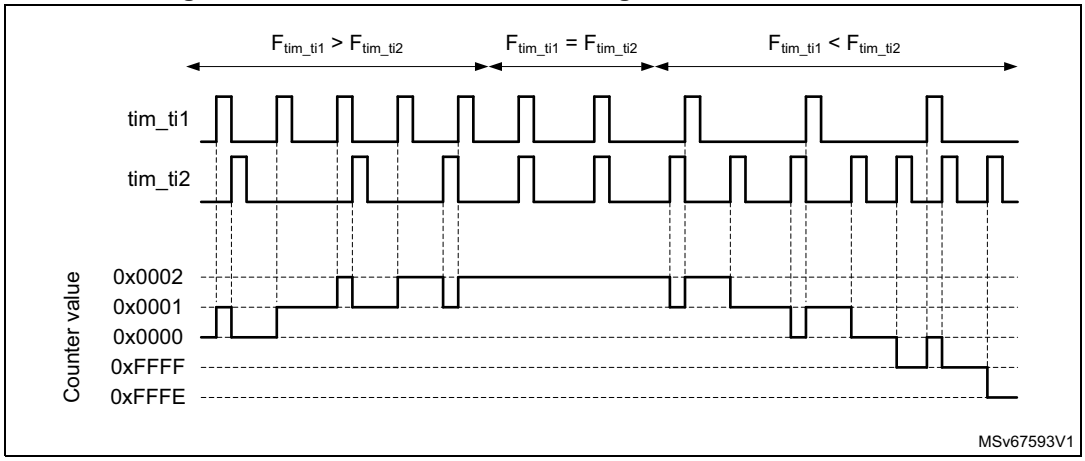
To do so, a signal representing the rate of a first audio stream must be connected to tim_ti1 and a signal representing the rate of a second audio stream must be connected to tim_ti2. The signals representing the audio rates are generally the SAIx_FS_A, SAIx_FS_B, SPIx_WS or spdifrx_frame_sync (this list of signals is provided as example and depends on the audio peripheral availability for a given product line, as detailed on the product datasheet).

In this case, the clocks are present on both tim_ti1 and tim2 simultaneously, as shown on *Figure 699*, and the counter value indicates the drift.

Let's define F_{tim_ti1} and F_{tim_ti2} the frequencies of the signals on tim_ti1 and tim_ti2. The counter value will change as following:

- If $F_{tim_ti1} > F_{tim_ti2}$ the counter value increases.
- If $F_{tim_ti1} < F_{tim_ti2}$ the counter value decreases.
- If $F_{tim_ti1} = F_{tim_ti2}$ the counter value will dither between two consecutive values, from one clock edge to the other. If the two clocks are perfectly synchronized (clock edges occurring simultaneously on the two inputs), the counter value does not change (as shown on *Figure 699*).

Figure 699. Clock drift measure using directional clock mode



35.4.24 UIF bit remapping

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIM_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

35.4.25 Timer input XOR function

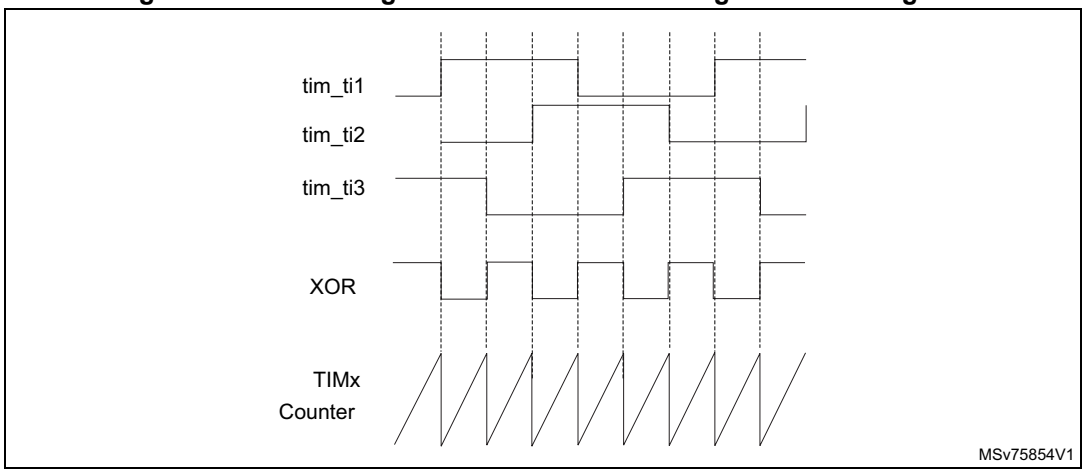
The TI1S bit in the TIM_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in *Section 34.3.34: Interfacing with Hall sensors*.

It is convenient to measure the interval between edges on two input signals, as per *Figure 700*.

Figure 700. Measuring time interval between edges on three signals



The XORPS control bit selects the position of the XOR gate, as shown on the *Figure 701* and *Figure 702* below, when the TI1S bit is set.

Figure 701. XOR gate connection with TI1S = 1 and XORPS = 0

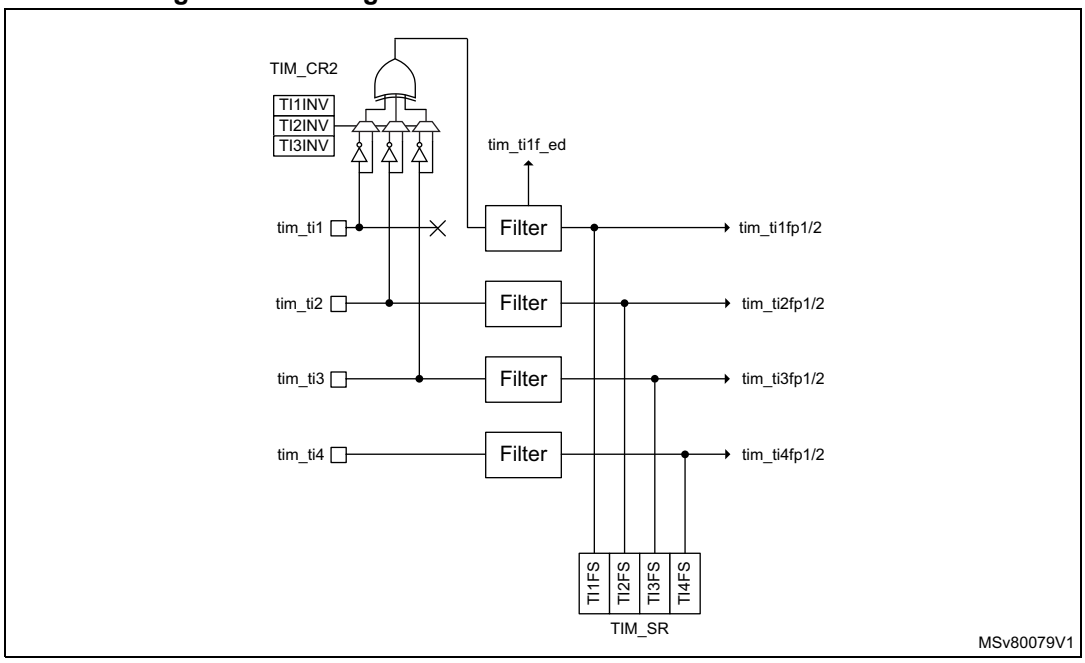
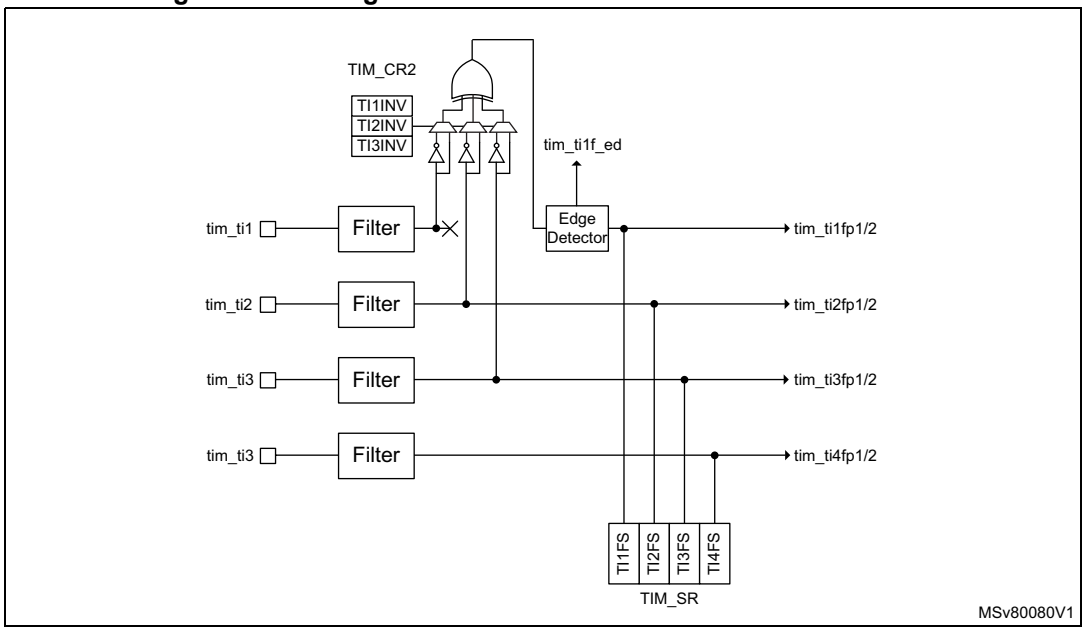


Figure 702. XOR gate connection with TI1S = 1 and XORPS = 1



Three control bits T1INV, T2INV and T3INV in the TIM_CR2 register can invert the tim_in1, tim_in2 and tim_in3 signals prior entering the XOR gate.

The status of the tim_in1, tim_in2, tim_in3 and tim_in4 signals can be software polled using the TI1FS, TI2FS, TI3FS and TI4FS bits in the TIM_SR register (tim_ti4 is not represented on figures above and do not enter into the XOR gate).

35.4.26 Timers and external trigger synchronization

The TIMx timers can be synchronized with an external trigger in several modes: reset mode, gated mode, trigger mode, reset + trigger and gated + reset modes.

Slave mode: reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIM_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIM_ARR, TIM_CCRx) are updated.

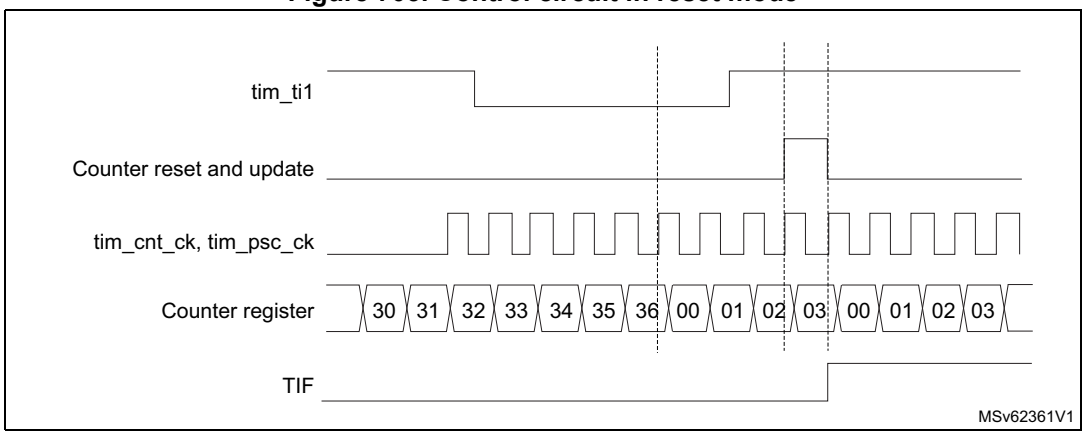
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIM_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 100 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Start the counter by writing CEN = 1 in the TIM_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIM_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIM_DIER register).

The following figure shows this behavior when the autoreload register TIM_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 703. Control circuit in reset mode



Slave mode: gated mode

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The

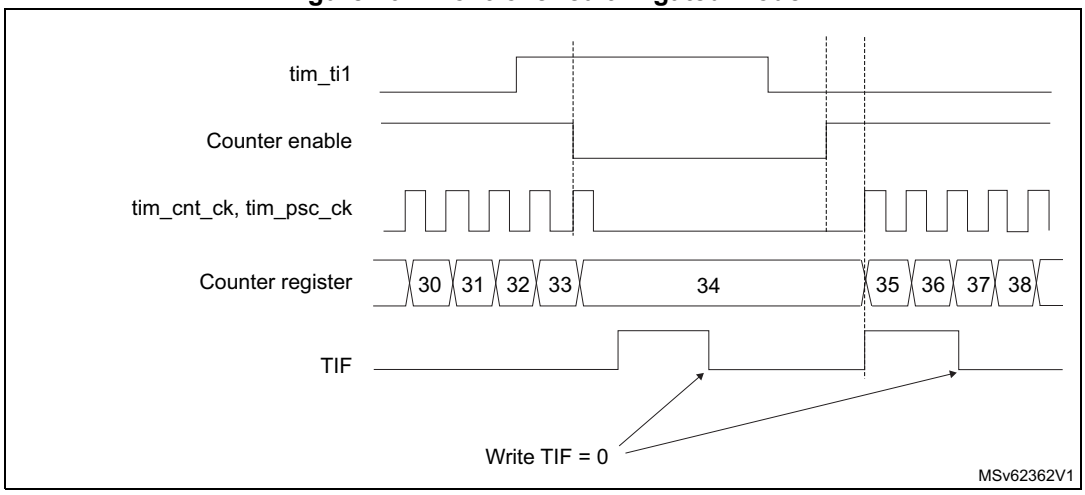
capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIM_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in gated mode by writing SMS = 101 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIM_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIM_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 704. Control circuit in gated mode



Note: The configuration "CCxP = CCxNP = 1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: trigger mode

The counter can start in response to an event on a selected input.

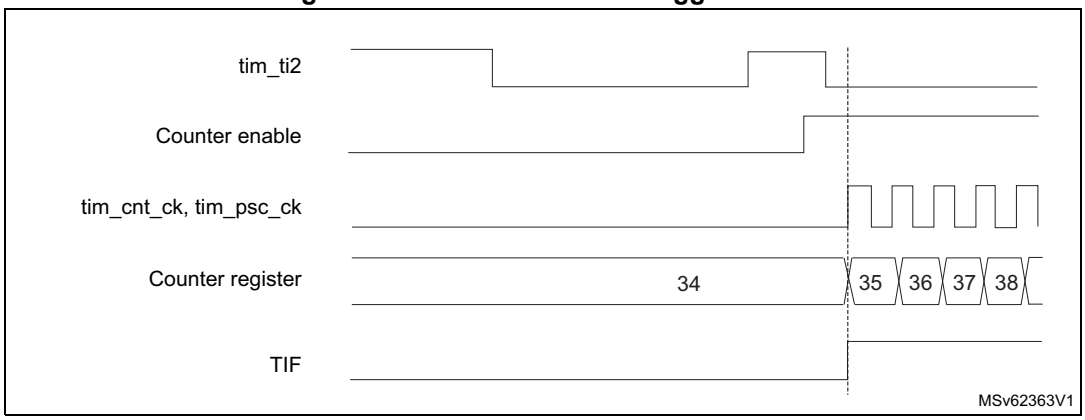
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S = 01 in TIM_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in TIM_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIM_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 705. Control circuit in trigger mode



Slave mode selection preload for run-time encoder mode update

The SMS[4:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIM_SMCR register. The trigger for the transfer from SMS[4:0] preload to active value is the update event (UEV) occurring when the counter overflows.

Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode – combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode – external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode, or trigger mode. It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIM_SMCR register.

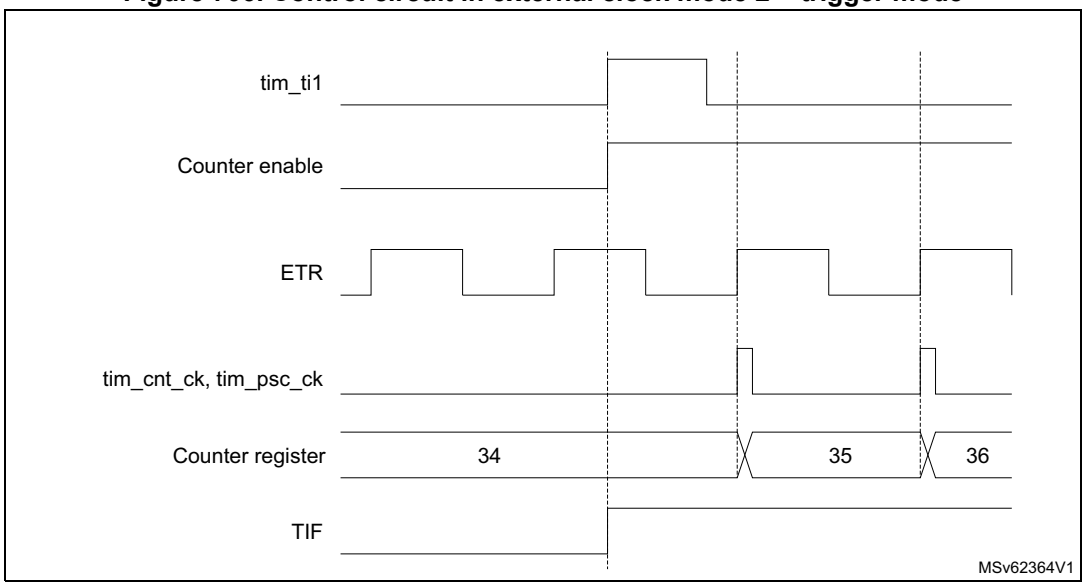
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIM_SMCR register as follows:
   - ETF = 0000: no filter.
   - ETPS = 00: prescaler disabled.
   - ETP = 0: detection of rising edges on tim_etr_in and ECE = 1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in TIM_CCMR1 register to select only the input capture source.
   - CC1P = 0 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS = 110 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 706. Control circuit in external clock mode 2 + trigger mode



35.4.27 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one timer is configured in master mode, it can reset, start, stop, or clock the counter of another timer configured in slave mode.

Figure 707 and *Figure 708* show examples of master/slave timer connections.

Figure 707. Master/Slave timer example

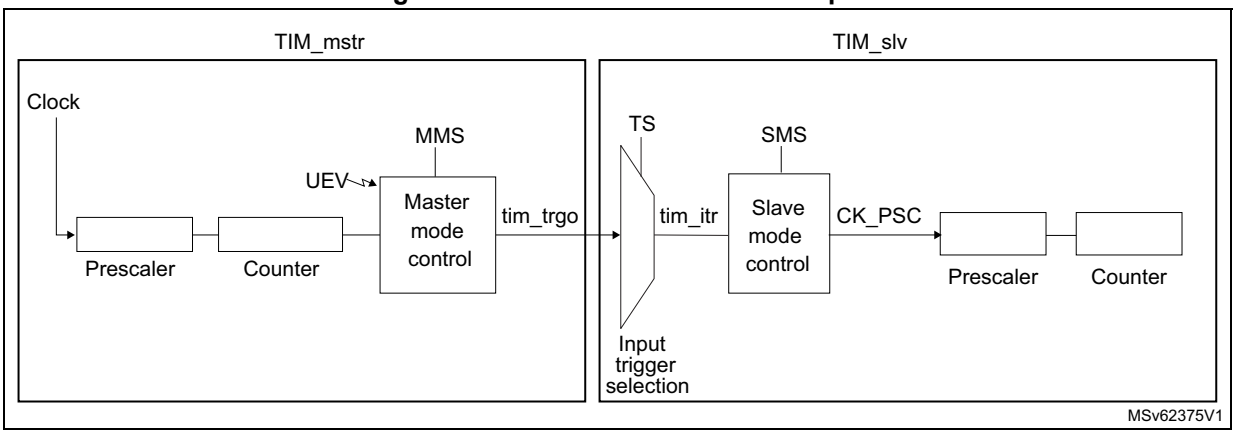
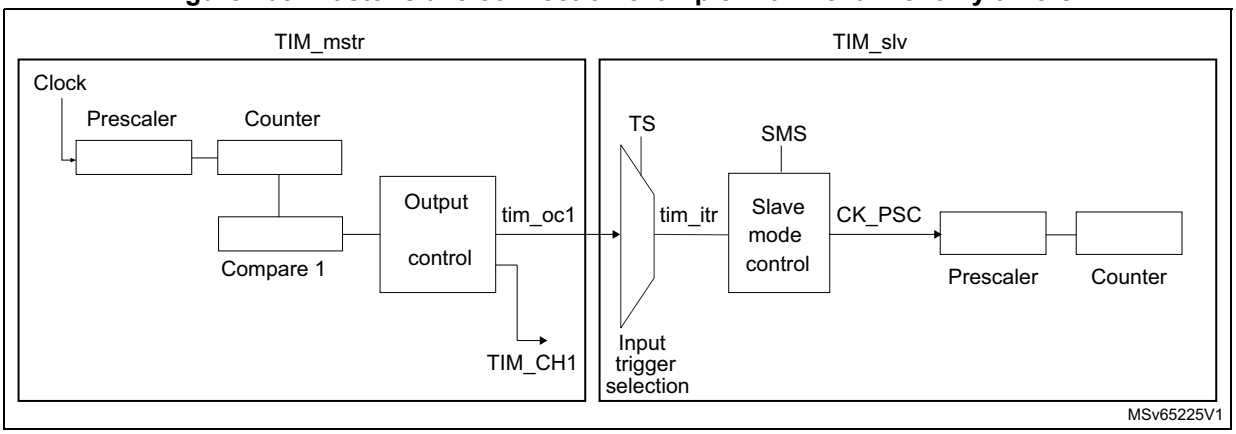


Figure 708. Master/slave connection example with 1 channel only timers



Note: The timers with one channel only (see Figure 708) do not feature a master mode. However, the tim_oc1 output signal can serve as trigger for slave timer (see TIMx internal trigger connection table in Section 35.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals). The tim_oc1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger. For instance, if the destination timer tim_ker_ck clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

Using one timer as prescaler for another timer

For example, TIM_mstr can be configured to act as a prescaler for TIM_slv. Refer to *Figure 707*. To do this:

1. Configure TIM_mstr in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS = 010 is written in the TIM_mstr_CR2 register, a rising edge is output on tim_trgo each time an update event is generated.
2. To connect the tim_trgo output of TIM_mstr to TIM_slv, TIM_slv must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM_slv_SMCR register (writing TS = 00010).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS = 111 in the TIM_slv_SMCR register). This causes TIM_slv to be clocked by the rising edge

of the periodic TIM_mstr trigger signal (which correspond to the TIM_mstr counter overflow).

4. Finally both timers must be enabled by setting their respective CEN bits (TIM_CR1 register).

Note: If tim_ocx is selected on TIM_mstr as the trigger output (MMS = 1xx), its rising edge is used to clock the counter of TIM_slv.

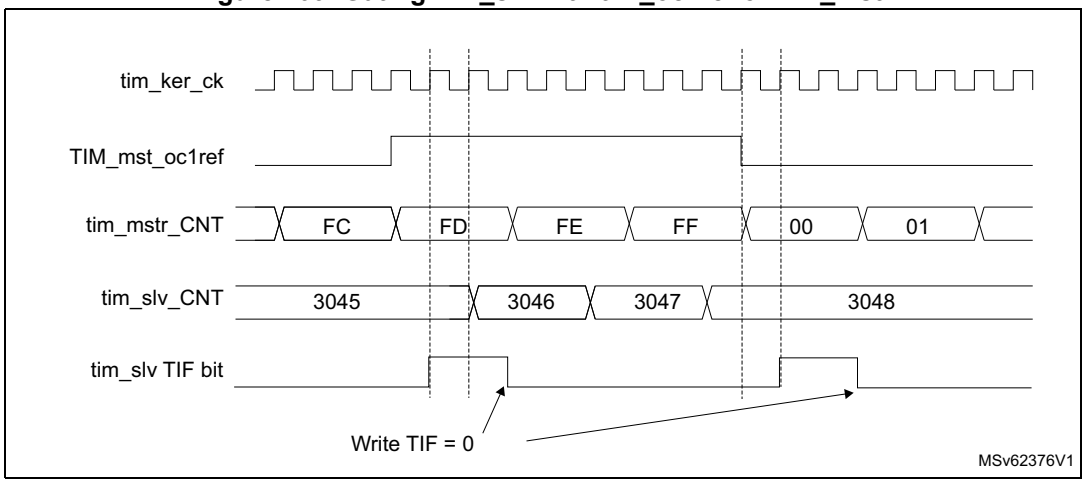
Using one timer to enable another timer

In this example, we control the enable of TIM_slv with the output compare 1 of TIM_mstr. Refer to *Figure 707* for connections. TIM_slv counts on the divided internal clock only when tim_oc1ref of TIM_mstr is high. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).
5. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).
6. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

Note: The slave timer counter clock is not synchronized with the master timer counter clock, this mode only affects the TIM_slv counter enable signal.

Figure 709. Gating TIM_slv with tim_oc1ref of TIM_mstr

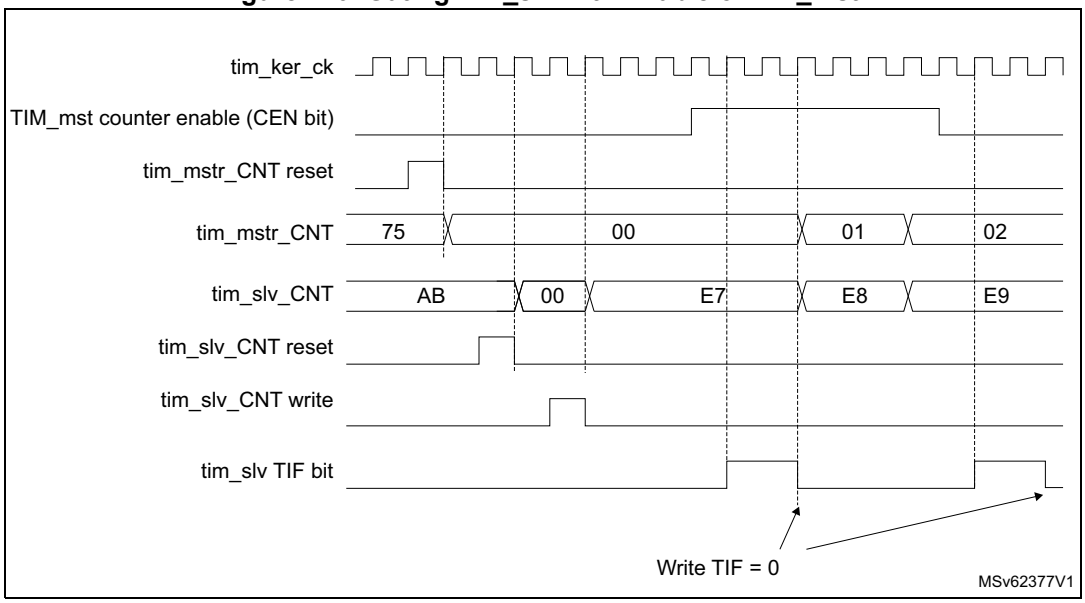


In the example in *Figure 709*, the TIM_slv counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM_mstr. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIM_EGR registers.

In the next example (refer to *Figure 710*), we synchronize TIM_mstr and TIM_slv. TIM_mstr is the master and starts from 0. TIM_slv is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM_slv stops when TIM_mstr is disabled by writing 0 to the CEN bit in the TIM_mstr_CR1 register:

1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).
5. Reset TIM_mstr by writing 1 in UG bit (TIM_mstr_EGR register).
6. Reset TIM_slv by writing 1 in UG bit (TIM_slv_EGR register).
7. Initialize TIM_slv to 0xE7 by writing 0xE7 in the TIM_slv counter (TIM_slv_CNT).
8. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).
9. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).
10. Stop TIM_mstr by writing 0 in the CEN bit (TIM_mstr_CR1 register).

Figure 710. Gating TIM_slv with Enable of TIM_mstr

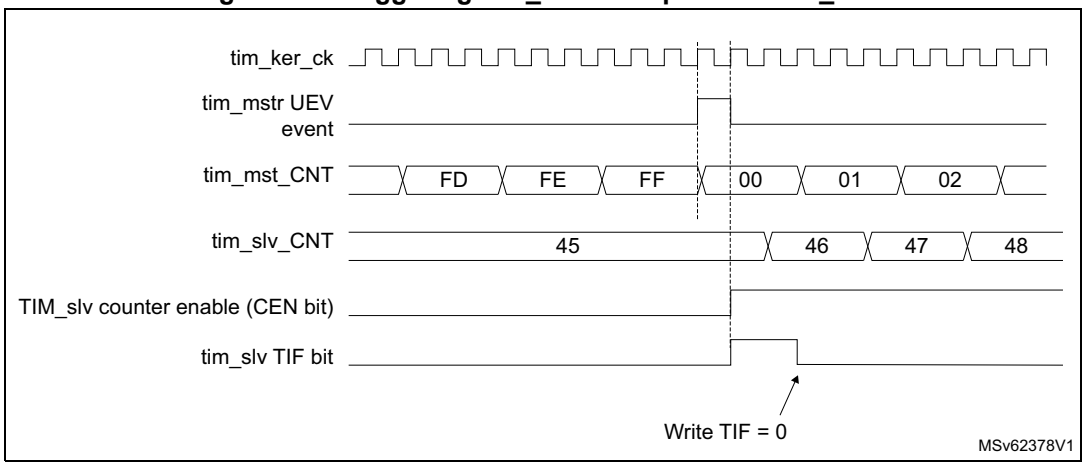


Using one timer to start another timer

In this example, we set the enable of TIM_slv with the update event of TIM_mstr. Refer to *Figure 707* for connections. TIM_slv starts counting from its current value (which can be nonzero) on the divided internal clock as soon as the update event is generated by TIM_mstr. When TIM_slv receives the trigger signal its CEN bit is automatically set and the counter counts until we write 0 to the CEN bit in the TIM_slv_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

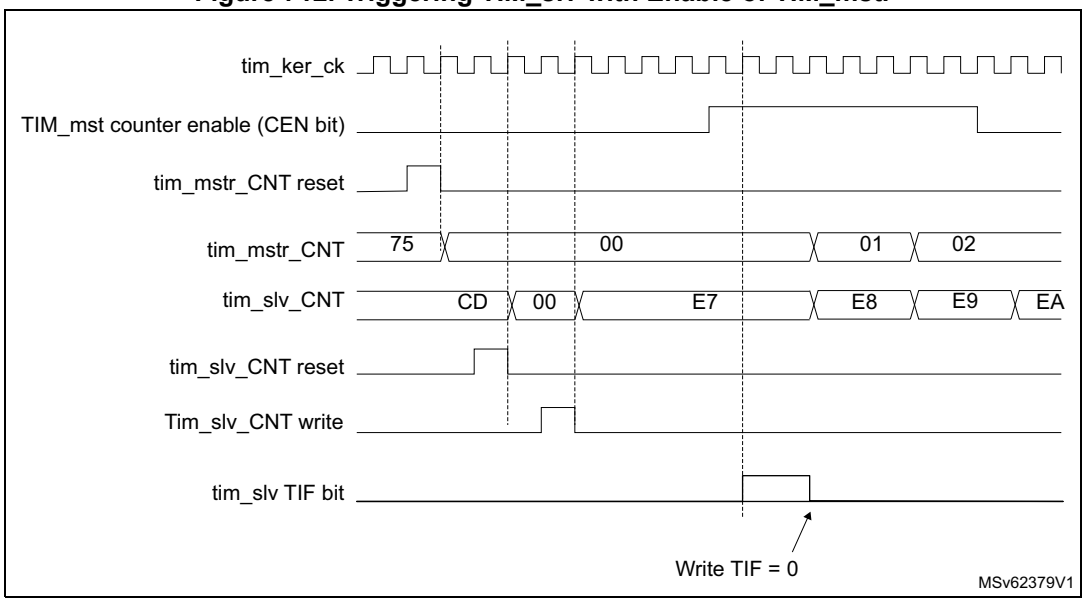
1. Configure TIM_mstr master mode to send its update event (UEV) as trigger output (MMS = 010 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr period (TIM_mstr_ARR registers).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in trigger mode (SMS = 110 in TIM_slv_SMCR register).
5. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

Figure 711. Triggering TIM_slv with update of TIM_mstr



As in the previous example, both counters can be initialized before starting counting. *Figure 712* shows the behavior with the same configuration as in *Figure 711* but in trigger mode (SMS = 110 in the TIM_slv_SMCR register) instead of gated mode.

Figure 712. Triggering TIM_slv with Enable of TIM_mstr



Starting two timers synchronously in response to an external trigger

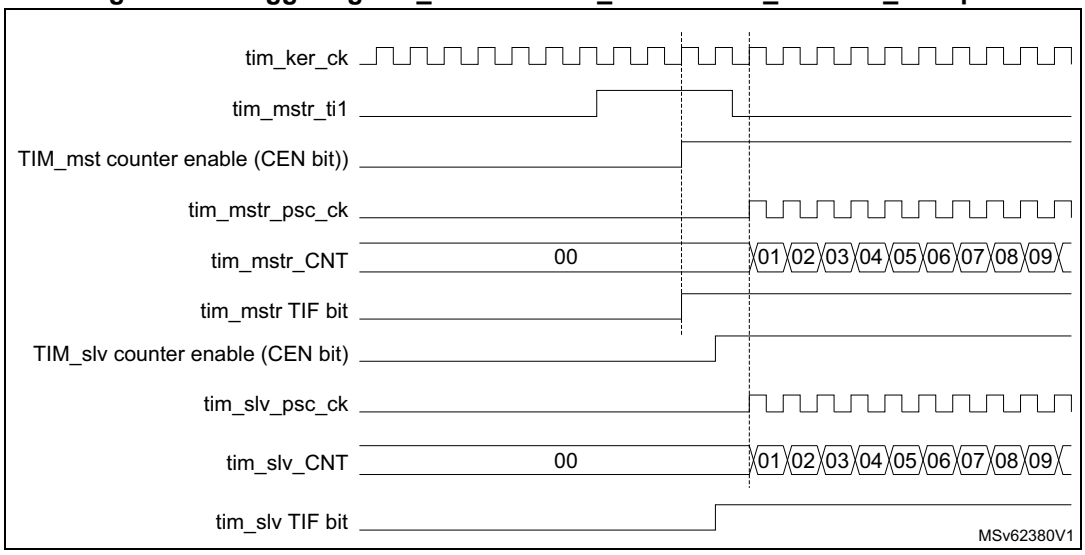
In this example, we set the enable of TIM_mstr when its tim_ti1 input rises, and the enable of TIM_slv with the enable of TIM_mstr. Refer to *Figure 707* for connections. To ensure the counters are aligned, TIM_mstr must be configured in master/slave mode (slave with respect to tim_ti1, master with respect to TIM_slv):

1. Configure TIM_mstr master mode to send its enable as trigger output (MMS = 001 in the TIM_mstr_CR2 register).
2. Configure TIM_mstr slave mode to get the input trigger from tim_ti1 (TS = 00100 in the TIM_mstr_SMCR register).
3. Configure TIM_mstr in trigger mode (SMS = 110 in the TIM_mstr_SMCR register).
4. Configure the TIM_mstr in master/slave mode by writing MSM = 1 (TIM_mstr_SMCR register).
5. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00000 in the TIM_slv_SMCR register).
6. Configure TIM_slv in trigger mode (SMS = 110 in the TIM_slv_SMCR register).

When a rising edge occurs on tim_ti1 (TIM_mstr), both counters start counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIM_CNT). One can see that the master/slave mode inserts a delay between CNT_EN and CK_PSC on TIM_mstr.

Figure 713. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input



Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

35.4.28 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

35.4.29 ADC synchronization

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIM_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- When the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.
- When the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source):

1. The counter period must be a multiple of the ADC clock period.

$$(\text{TIM_PSC} + 1) \times (\text{TIM_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period.

$$(\text{TIM_PSC} + 1) \times \text{TIM_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements is met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock.
2. Configure the timer and set the ADSYNC bit.
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits).
4. Start the timer (with the CEN counter enable bit).

35.4.30 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to reprogram part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIM_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIM_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIM_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIM_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register:

Example:

00000: TIM_CR1

00001: TIM_CR2

00010: TIM_SMCR

The DBSS[3:0] bits in the TIM_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 35.5.23: TIM DMA control register (TIM_DCR)* for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address.
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bitfields as follows: DBL = 3, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx.
5. Enable the DMA channel.

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

35.4.31 TIM2/TIM3/TIM4/TIM5 DMA requests

The TIM2/TIM3/TIM4/TIM5 can generate a DMA requests, as shown in *Table 408*.

Table 408. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_cc3_dma	Capture/compare 3	CC3DE
tim_cc4_dma	Capture/compare 4	CC4DE
tim_trgi_dma	Trigger	TDE

Note: Some timer's DMA requests may not be connected to the DMA controller. Refer to the DMA section(s) for more details.

35.4.32 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter can either continue to work normally or stops.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

35.4.33 TIM2/TIM3/TIM4/TIM5 low-power modes

Table 409. Effect of low-power modes on TIM2/TIM3/TIM4/TIM5

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

35.4.34 TIM2/TIM3/TIM4/TIM5 interrupts

The TIM2/TIM3/TIM4/TIM5 can generate multiple interrupts, as shown in *Table 410*.

Table 410. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UPD	Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
	Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRGI	Trigger	TIF	TIE	write 0 in TIF	Yes	No
TIM_DIR_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
TIM_TERR	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

35.5 TIM2/TIM3/TIM4/TIM5 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

35.5.1 TIM control register 1 (TIM_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CKD2[3:0]**: Clock division 2

This bit field indicates the division between sampling clock (tDTS) used by the dead-time generators and the digital filters (tim_etr_in, tim_tix).

0000: No Division
0001: $t_{DTS2} = 4*t_{DTS}$
0010: $t_{DTS2} = 16*t_{DTS}$
0011: $t_{DTS2} = 64*t_{DTS}$
0100: $t_{DTS2} = 256*t_{DTS}$
0101: $t_{DTS2} = 1024*t_{DTS}$
0110: $t_{DTS2} = 4096*t_{DTS}$
0111: $t_{DTS2} = 16384*t_{DTS}$
1000: $t_{DTS2} = 65536*t_{DTS}$
1001: $t_{DTS2} = 262144*t_{DTS}$
Others: Reserved, do not program this value

Bits 23:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering Enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_etr_in, tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2 \times t_{tim_ker_ck}$
10: $t_{DTS} = 4 \times t_{tim_ker_ck}$
11: $t_{DTS} = 8 \times t_{tim_ker_ck}$

Bit 7 **ARPE**: Autoreload preload enable

0: TIM_ARR register is not buffered
1: TIM_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in center-aligned mode or encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

35.5.2 TIM control register 2 (TIM_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TI3INV	TI2INV	TI1INV	AD SYNC	XORPS	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw		rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bit 31 **TI3INV**: tim_ti3 signal inversion on XOR gate input
0: tim_ti3 non-inverted
1: tim_ti3 inverted

Bit 30 **TI2INV**: tim_ti2 signal inversion on XOR gate input
0: tim_ti2 non-inverted
1: tim_ti2 inverted

Bit 29 **TI1INV**: tim_ti1 signal inversion on XOR gate input
0: tim_ti1 non-inverted
1: tim_ti1 inverted

Bit 28 **ADSYNC**: ADC synchronization
0: The timer operates independently from the ADC
1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 35.4.29* for requirements.
The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bit 27 **XORPS**: XOR gate position selection
This bit defines the position of the XOR gate with respect to input filters.
0: XOR gate placed before TI1 filter
1: XOR gate placed after TI1, TI2 and TI3 filters

Bit 26 Reserved, must be kept at reset value.

Bits 24:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: tim_ti1 selection
This bit enables the insertion of a XOR gate to combine tim_ti1_in[15:0], tim_ti2_in[15:0], and tim_ti3_in[15:0] inputs into tim_ti1.
The position of the XOR gate, with respect to input filters, is defined with the XORPS bit.
0: The tim_ti1_in[15:0] multiplexer output is to tim_ti1 input
1: The tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input. See also *Section 34.3.34: Interfacing with Hall sensors*.

Bits 25, 6, 5, 4 **MMS[3:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIM_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

Others: Reserved

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, must be kept at reset value.

35.5.3 TIM slave mode control register (TIM_SMCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SETPS[3:0]				SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	SMS[4:3]	
		rw	rw	rw	rw	rw	rw			rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:26 **SETPS[3:0]**: Synchronous external trigger prescaler

The linear synchronous prescaler divides the frequency of the tim_etrf signal, by a factor (SETPS[3:0] + 1).

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[4:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[4:0] bitfield is preloaded

0: SMS[4:0] bitfield is not preloaded

1: SMS[4:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:18 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or $\overline{\text{tim_etr_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge

1: tim_etr_in is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables external clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS = 111 and TS = 00111).
It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).
If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of tim_ker_ck frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF
01: tim_etrp frequency divided by 2
10: tim_etrp frequency divided by 4
11: tim_etrp frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bitfield then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bit 7 **MSM**: Master/slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal trigger 0 (tim_itr0)
00001: Internal trigger 1 (tim_itr1)
00010: Internal trigger 2 (tim_itr2)
00011: Internal trigger 3 (tim_itr3)
00100: tim_ti1 edge detector (tim_ti1f_ed)
00101: Filtered timer input 1 (tim_ti1fp1)
00110: Filtered timer input 2 (tim_ti2fp2)
00111: External trigger input (tim_etrf)
01000: Internal trigger 4 (tim_itr4)
01001: Internal trigger 5 (tim_itr5)
01010: Internal trigger 6 (tim_itr6)
01011: Internal trigger 7 (tim_itr7)
01100: Internal trigger 8 (tim_itr8)
01101: Internal trigger 9 (tim_itr9)
01110: Internal trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
10100: Internal trigger 16 (tim_itr16)
10101: Internal trigger 17 (tim_itr17)
10110: Internal trigger 18 (tim_itr18)
10111: Internal trigger 19 (tim_itr19)
11000: Internal trigger 20 (tim_itr20)
11001: Internal trigger 21 (tim_itr21)
11010: Internal trigger 22 (tim_itr22)
11011: Internal trigger 23 (tim_itr23)
11100: Internal trigger 24 (tim_itr24)
11101: Internal trigger 25 (tim_itr25)
11110: Internal trigger 26 (tim_itr26)
11111: Internal trigger 27 (tim_itr27)

See *Section 35.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation details.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source

0: tim_ocref_clr_int is connected to the tim_ocref_clr input

1: tim_ocref_clr_int is connected to tim_etrf

Note: If the OCREF clear selection feature is not supported, this bit is reserved and forced by hardware to 0. Section 35.3: TIM2/TIM3/TIM4/TIM5 implementation.

Bits 17:16, 2:0 **SMS[4:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIM_SMCR for tim_etr_in and CCxP/CCxNP bits in TIM_CCER register for tim_ti1fp1 and tim_ti2fp2).

00000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

00001: Encoder mode 1 - Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

00010: Encoder mode 2 - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

00011: Encoder mode 3 - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

00100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

00101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

00110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

00111: External clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

01000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

01001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

01010: Encoder mode: Clock plus direction, x2 mode.

01011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P.

01100: Encoder mode: Directional clock, x2 mode.

01101: Encoder mode: Directional clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

01110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

01111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

10000: Quadrature encoder with built-in debouncer: x2 mode.

10001: Quadrature encoder with built-in debouncer: x4 mode.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

35.5.4 TIM DMA/interrupt enable register (TIM_DIER)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction change interrupt disabled
1: Direction change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled.
1: Trigger DMA request enabled.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CC4DE**: Capture/compare 4 DMA request enable
0: CC4 DMA request disabled.
1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/compare 3 DMA request enable
0: CC3 DMA request disabled.
1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled.
1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled.
1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/compare 4 interrupt enable
0: CC4 interrupt disabled.
1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/compare 3 interrupt enable
0: CC3 interrupt disabled.
1: CC3 interrupt enabled.

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

35.5.5 TIM status register (TIM_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TI4FS	TI3FS	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.
r	r	r	r				rc_w0	rc_w0	rc_w0	rc_w0	rc_w0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bit 31 **TI4FS**: tim_ti4f status
This flag indicates the value of the tim_ti4f signal (after the digital filtering stage), for polling purpose.
0: tim_ti4f signal is low
1: tim_ti4f signal is high

Bit 30 **TI3FS**: tim_ti3f status
This flag indicates the value of the tim_ti3f signal (after the digital filtering stage), for polling purpose.
0: tim_ti3f signal is low
1: tim_ti3f signal is high

Bit 29 **TI2FS**: tim_ti2f status
This flag indicates the value of the tim_ti2f signal (after the digital filtering stage), for polling purpose.
0: tim_ti2f signal is low
1: tim_ti2f signal is high

Bit 28 **TI1FS**: tim_ti1f status
This flag indicates the value of the tim_ti1f signal (after the digital filtering stage), for polling purpose.
0: tim_ti1f signal is low
1: tim_ti1f signal is high

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **UIOVRF**: Update Interrupt Overrun flag
This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.
0: No Update Interrupt Overrun
1: Update Interrupt Overrun

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to 0.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIM_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to 0.
0: No index event occurred.
1: An index event has occurred

Bits 19:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIM_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input) when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/compare 4 interrupt flag
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/compare 3 interrupt flag
Refer to CC1IF description

Bit 2 **CC2IF**: Capture/compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIM_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIM_CNT matches the content of the TIM_CCR1 register. When the content of TIM_CCR1 is greater than the content of TIM_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are three possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIM_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIM_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIM_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
At overflow or underflow and if UDIS = 0 in the TIM_CR1 register.
When CNT is reinitialized by software using the UG bit in TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.
When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS = 0 and UDIS = 0 in the TIM_CR1 register.

35.5.6 TIM event generation register (TIM_EGR)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

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

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIM_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIM_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (up-counting), else it takes the autoreload value (TIM_ARR) if DIR = 1 (down-counting).

35.5.7 TIM capture/compare mode register 1 (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1.

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000:No filter, sampling is done at f_{DTS}

0001:$f_{SAMPLING} = f_{tim_ker_ck}$, N = 2

0010:$f_{SAMPLING} = f_{tim_ker_ck}$, N = 4

0011:$f_{SAMPLING} = f_{tim_ker_ck}$, N = 8

0100:$f_{SAMPLING} = f_{DTS}/2$, N = 6

0101:$f_{SAMPLING} = f_{DTS}/2$, N = 8

0110:$f_{SAMPLING} = f_{DTS}/4$, N = 6

0111:$f_{SAMPLING} = f_{DTS}/4$, N = 8

1000:$f_{SAMPLING} = f_{DTS}/8$, N = 6

1001:$f_{SAMPLING} = f_{DTS}/8$, N = 8

1010:$f_{SAMPLING} = f_{DTS}/16$, N = 5

1011:$f_{SAMPLING} = f_{DTS}/16$, N = 6

1100:$f_{SAMPLING} = f_{DTS}/16$, N = 8

1101:$f_{SAMPLING} = f_{DTS}/32$, N = 5

1110:$f_{SAMPLING} = f_{DTS}/32$, N = 6

1111:$f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIM_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

35.5.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC2M [4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M [4:3]	
						rw	rw							rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

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

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

0: tim_oc2ref is not affected by the tim_ocref_clr_int signal

1: tim_oc2ref is cleared as soon as a high level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 35.4.16: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 25:24, 14:12 **OC2M[4:0]**: Output compare 2 mode

Refer to OC1M description on bits 17:16, 6:4.

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int input

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input

Note: This mode is not available for all PWM modes. Refer to Section 35.4.16: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC1M[4:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

00000: Frozen - The comparison between the output compare register TIM_CCR1 and the counter TIM_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

00001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00011: Toggle - tim_oc1ref toggles when TIM_CNT = TIM_CCR1.

00100: Force inactive level - tim_oc1ref is forced low.

00101: Force active level - tim_oc1ref is forced high.

00110: PWM mode 1 - In up-counting, channel 1 is active as long as TIM_CNT<TIM_CCR1 else inactive. In down-counting, channel 1 is inactive (tim_oc1ref = 0) as long as TIM_CNT>TIM_CCR1 else active (tim_oc1ref = 1).

00111: PWM mode 2 - In up-counting, channel 1 is inactive as long as TIM_CNT<TIM_CCR1 else active. In down-counting, channel 1 is active as long as TIM_CNT>TIM_CCR1 else inactive.

01000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01010: Reserved.

01011: Reserved.

01100-01101: Combined PWM mode 1 and 2 - Refer to *Table 405: Combined PWM mode overview* for details.

01110-01111: Asymmetric PWM mode 1 and 2 - Refer to *Table 404: Asymmetric PWM modes* for details.

Others: Reserved.

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIM_CCR1 disabled. TIM_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIM_CCR1 enabled. Read/Write operations access the preload register. TIM_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to three clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

35.5.9 TIM capture/compare mode register 2 (TIM_CCMR2)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIM_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIM_CCER).

35.5.10 TIM capture/compare mode register 2 [alternate] (TIM_CCMR2)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC4M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC3M[4:3]	
						rw	rw							rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

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

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable
0: tim_oc4ref is not affected by the tim_ocref_clr_int signal
1: tim_oc4ref is cleared as soon as a high level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)
Note: This mode is not available for all PWM modes. Refer to Section 35.4.16: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 25:24, 14:12 **OC4M[4:0]**: Output compare 4 mode
Refer to OC3M[4:0].

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4

10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3

11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIM_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

0: tim_oc3ref is not affected by the tim_ocref_clr_int signal

1: tim_oc3ref is cleared as soon as a high level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 35.4.16: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC3M[4:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

00000: Frozen - The comparison between the output compare register TIM_CCR3 and the counter TIM_CNT has no effect on the outputs.(this mode is used to generate a timing base).

00001: Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIM_CNT matches the capture/compare register 3 (TIM_CCR3).

00010: Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIM_CNT matches the capture/compare register 3 (TIM_CCR3).

00011: Toggle - tim_oc3ref toggles when TIM_CNT = TIM_CCR3.

00100: Force inactive level - tim_oc3ref is forced low.

00101: Force active level - tim_oc3ref is forced high.

00110: PWM mode 1 - In up-counting, channel 3 is active as long as TIM_CNT<TIM_CCR3 else inactive. In down-counting, channel 3 is inactive (tim_oc3ref = 0) as long as TIM_CNT>TIM_CCR3 else active (tim_oc3ref = 1).

00111: PWM mode 2 - In up-counting, channel 3 is inactive as long as TIM_CNT<TIM_CCR3 else active. In down-counting, channel 3 is active as long as TIM_CNT>TIM_CCR3 else inactive.

01000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01010: Pulse on compare - A pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIM_ECR.

01011: Direction output - The tim_oc3ref signal is overridden by a copy of the DIR bit.

01100-01101: Combined PWM mode 1 and 2 - Refer to *Table 405: Combined PWM mode overview* for details.

01110-01111: Asymmetric PWM mode 1 and 2 - Refer to *Table 404: Asymmetric PWM modes* for details.

10110-11001: Asymmetric PWM mode 7 to 10 - Available on OC3M only. Refer to *Table 404: Asymmetric PWM modes* for details.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIM_CCER).

35.5.11 TIM capture/compare enable register (TIM_CCER)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bit 15 **CC4NP**: Capture/compare 4 output Polarity.
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/compare 4 output Polarity.
Refer to CC1P description

Bit 12 **CC4E**: Capture/compare 4 output enable.
refer to CC1E description

Bit 11 **CC3NP**: Capture/compare 3 output Polarity.
Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: Capture/compare 3 output Polarity.
Refer to CC1P description

Bit 8 **CC3E**: Capture/compare 3 output enable.
Refer to CC1E description

Bit 7 **CC2NP**: Capture/compare 2 output Polarity.
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/compare 2 output Polarity.
refer to CC1P description

Bit 4 **CC2E**: Capture/compare 2 output enable.
Refer to CC1E description

Bit 3 **CC1NP**: Capture/compare 1 output Polarity.

CC1 channel configured as output: CC1NP must be kept cleared in this case.

CC1 channel configured as input: This bit is used in conjunction with CC1P to define tim_ti1fp1/tim_ti2fp1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/compare 1 output Polarity.

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0:non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1:inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1:non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0:this configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/compare 1 output enable.

0: Capture mode disabled / OC1 is not active

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

Table 411. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The state of the external IO pins connected to the standard tim_ocx channels depends only on the GPIO registers when CCxE = 0.

35.5.12 TIM counter (TIM_CNT)

Address offset: 0x024

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY_CNT[31]**: Value depends on IUFREMAP in TIM_CR1.
If UIFREMAP = 0
CNT[31]: Most significant bit of counter value
If UIFREMAP = 1
UIFCPY: UIF Copy
This bit is a read-only copy of the UIF bit of the TIM_ISR register

Bits 30:0 **CNT[30:0]**: Least significant part of counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register holds the non-dithered part in CNT[30:0]. The fractional part is not available.

35.5.13 TIM prescaler (TIM_PSC)

Address offset: 0x028

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM_EGR register or through trigger controller when configured in “reset mode”).

35.5.14 TIM autoreload register (TIM_ARR)

Address offset: 0x02C

Reset value: 0xFFFF FFFF

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

Bits 31:0 **ARR[31:0]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 35.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the autoreload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the autoreload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[31:4]. The ARR[3:0] bitfield contains the dithered part.

35.5.15 TIM capture/compare register 1 (TIM_CCR1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR1[31:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[31:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIM_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[31:0]. The CCR1[3:0] bits are reset.

35.5.16 TIM capture/compare register 2 (TIM_CCR2)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR2[31:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIM_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[31:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIM_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[31:0]. The CCR2[3:0] bits are reset.

35.5.17 TIM capture/compare register 3 (TIM_CCR3)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR3[31:0]**: Capture/compare 3 value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIM_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[31:4]. The CCR3[3:0] bitfield contains the dithered part.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIM_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[31:0]. The CCR3[3:0] bits are reset.

35.5.18 TIM capture/compare register 4 (TIM_CCR4)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR4[31:0]**: Capture/compare 4 value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIM_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[31:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIM_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[31:0]. The CCR4[3:0] bits are reset.

35.5.19 TIM timer encoder control register (TIM_ECR)

Address offset: 0x058

Reset value: 0x0000 0000

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

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

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[4:0] = 00001, 00010, 00011, 01110, 01111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[4:0] = 01010, 01011, 01100, 01101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 **IBLK[1:0]**: Index blanking

This bit indicates if the Index event is conditioned by the tim_ti3 input

00: Index always active
01: Index disabled hen tim_ti3 input is active, as per CC3P bitfield
10: Index disabled when tim_ti4 input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction

This bit indicates in which direction the Index event resets the counter.

00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

35.5.20 TIM timer input selection register (TIM_TISEL)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 35.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

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

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIMx_CH3
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 35.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

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

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 35.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 35.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

35.5.21 TIM alternate function register 1 (TIM_AF1)

Address offset: 0x060

Reset value: 0x0000 0000

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:14 **ETRSEL[4:0]**: etr_in source selection
These bits select the etr_in input source.
00000: tim_etr0: TIM_ETR input
00001: tim_etr1
...
11111: tim_etr31
Refer to *Section 35.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 13:0 Reserved, must be kept at reset value.

35.5.22 TIM alternate function register 2 (TIM_AF2)

Address offset: 0x064

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **OCRSEL[3:0]**: ocref_clr source selection
These bits select the ocref_clr input source.
0000: tim_ocref_clr0
0001: tim_ocref_clr1
...
0111: tim_ocref_clr15
Refer to *Section 35.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 15:0 Reserved, must be kept at reset value.

35.5.23 TIM DMA control register (TIM_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address).

0000: Reserved
0001: Update
0010: CC1
0011: CC2
0100: CC3
0101: CC4
0110: COM
0111: Trigger
Others: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11111: 32 transfers

Example: Let us consider the following transfer: DBL = 6 bytes & DBA = TIM2_CR1.

–If DBL = 6 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation: (TIM_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIM_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIM_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register.

Example:
00000: TIM_CR1,
00001: TIM_CR2,
00010: TIM_SMCR,
...

35.5.24 TIM DMA address for full transfer (TIM_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIM_CR1 address) + (DBA + DMA index) x 4

where TIM_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM_DCR).

35.5.25 TIM2/TIM3/TIM4/TIM5 register map

TIM2/TIM3/TIM4/TIM5 registers are mapped as described in the table below.

Table 412. TIM2/TIM3/TIM4/TIM5 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**TIM_CR1**	Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value					0	0	0	0												0	0		0	0	0	0	0	0	0	0	0	0
0x004	**TIM_CR2**	TI3INV	TI2INV	TI1INV	ADSYNC	XORPS	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
	Reset value	0	0	0	0	0		0																		0	0	0	0	0			
0x008	**TIM_SMCR**	Res.	Res.	SETPS[3:0]				SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	SMS [4:3]		ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
	Reset value			0	0	0	0	0	0			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	**TIM_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value									0	0	0	0						0		0	0	0	0	0	0	0		0	0	0	0	0
0x010	**TIM_SR**	TI4FS	TI3FS	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value	0	0	0	0				0	0	0	0	0								0	0	0	0			0		0	0	0	0	0
0x014	**TIM_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x018	**TIM_CCMR1** Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR1 Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]		OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value							0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 412. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	**TIM_CCMR2** Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR2 Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC3M[4:3]		O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value							0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**TIM_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0
0x024	**TIM_CNT**	UIFCPY_CNT[31]	CNT[30:16] (CNT[31:16] on 32-bit timers only)															CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**TIM_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	**TIM_ARR**	ARR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	Reserved																																
0x034	**TIM_CCR1**	CCR1[31:20] (32-bit timers only)												CCR1[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	**TIM_CCR2**	CCR2[31:20] (32-bit timers only)												CCR2[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	**TIM_CCR3**	CCR3[31:20] (32-bit timers only)												CCR3[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	**TIM_CCR4**	CCR4[31:20] (32-bit timers only)												CCR4[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044.. 0x054	Reserved	Res.																															
0x058	**TIM_ECR**	Res.	Res.	Res.	Res.	Res.	PWPRSC [2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	IBLK [1:0]		IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x05C	**TIM_TISEL**	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	**TIM_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [4:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0	0	0														

Table 412. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x064	**TIM_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	0																
0x068.. 0x3D8	Reserved	Res.																															
0x3DC	**TIM_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	**TIM_DMAR**	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

36 Basic timers (TIM6/TIM7)

36.1 TIM6/TIM7 introduction

The basic timers TIM6/TIM7 consist in a 16-bit autoreload counter driven by a programmable prescaler.

They can be used as generic timers for time-base generation.

The basic timer can also be used for triggering the digital-to-analog converter. This is done with the trigger output of the timer.

The timers are completely independent, and do not share any resources.

36.2 TIM6/TIM7 main features

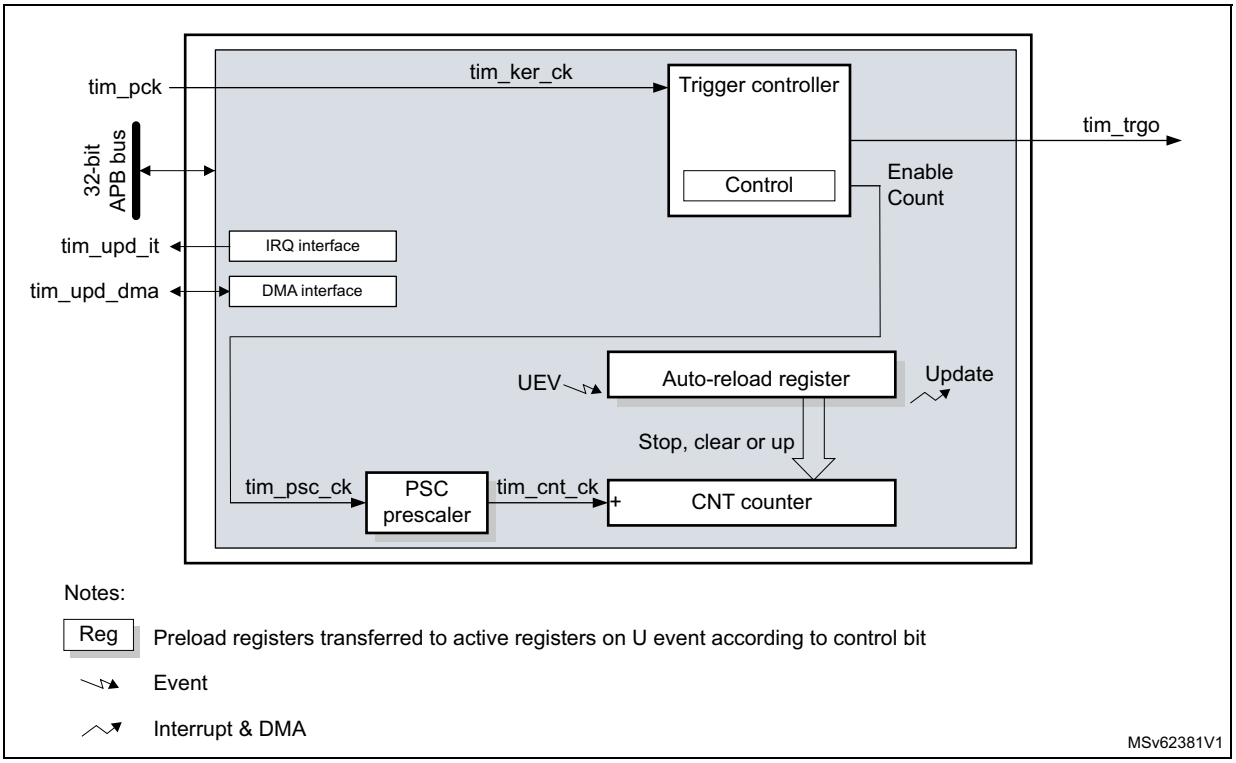
Basic timer (TIM6/TIM7) features include:

- 16-bit autoreload upcounter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Synchronization circuit to trigger the DAC.
- Interrupt/DMA generation on the update event: counter overflow.
- ADC synchronization for jitter-free sampling points.

36.3 TIM6/TIM7 functional description

36.3.1 TIM6/TIM7 block diagram

Figure 714. Basic timer block diagram



36.3.2 TIM6/TIM7 internal signals

The table in this section summarizes the TIM inputs and outputs.

Table 413. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer: 1, 2, 3,..., 16 (maximum value)
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals (DAC).
tim_upd_it	Output	Timer update event interrupt
tim_upd_dma	Output	Timer update dma request

36.3.3 TIM6/TIM7 clocks

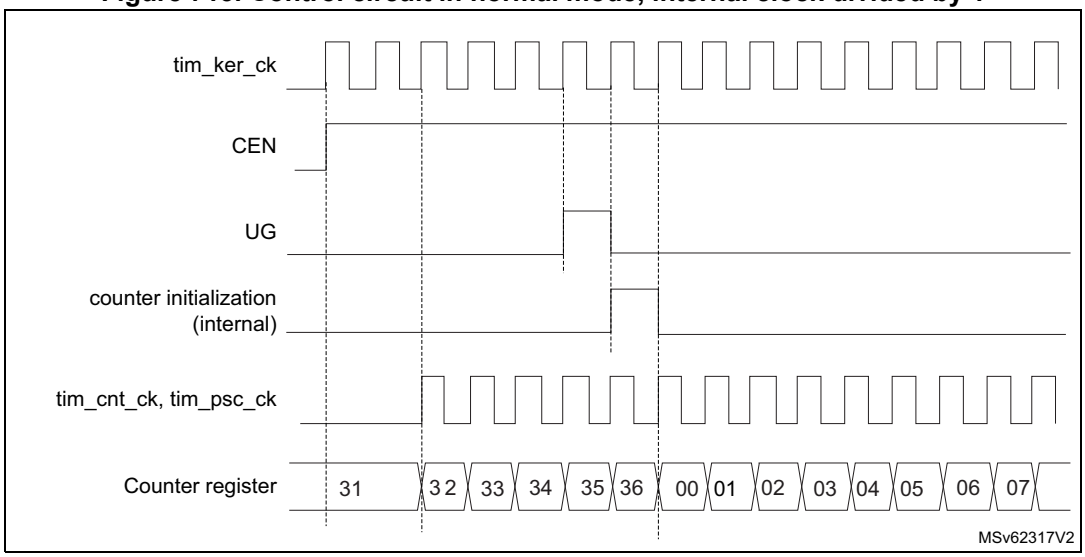
The timer bus interface is clocked by the tim_pclk APB clock.

The counter clock tim_ker_ck is connected to the tim_pclk input.

The CEN (in the TIM_CR1 register) and UG bits (in the TIM_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 715 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 715. Control circuit in normal mode, internal clock divided by 1



36.3.4 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIM_CNT)
- Prescaler register (TIM_PSC)
- Auto-Reload register (TIM_ARR).

The autoreload register is preloaded. The preload register is accessed each time an attempt is made to write or read the autoreload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the autoreload preload enable bit (ARPE) in the TIM_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIM_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

Note: An update overrun status bit (UIOVRF in TIM_SR register) indicates that an update request occurred while an update interrupt is pending (update interrupt flag set). This bit is useful when debugging closed loop applications, to indicate that:

- the update interrupt service routine is too long with respect to the interval between consecutive update events

- the update interrupt service routine was served too late with respect to the subsequent update event

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in the TIM_CR1 register is set.

Note that the actual counter enable signal tim_cnt_en is set one clock cycle after CEN bit set.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIM_PSC register). It can be changed on the fly as the TIM_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 716 and *Figure 717* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 716. Counter timing diagram with prescaler division change from 1 to 2

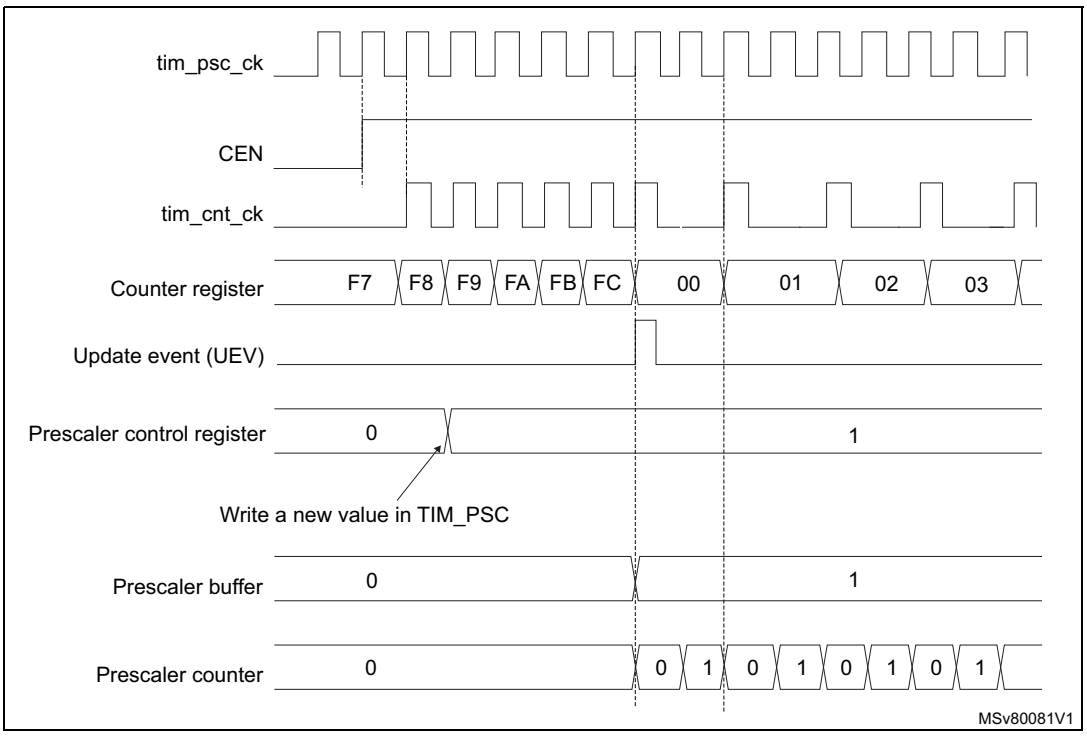
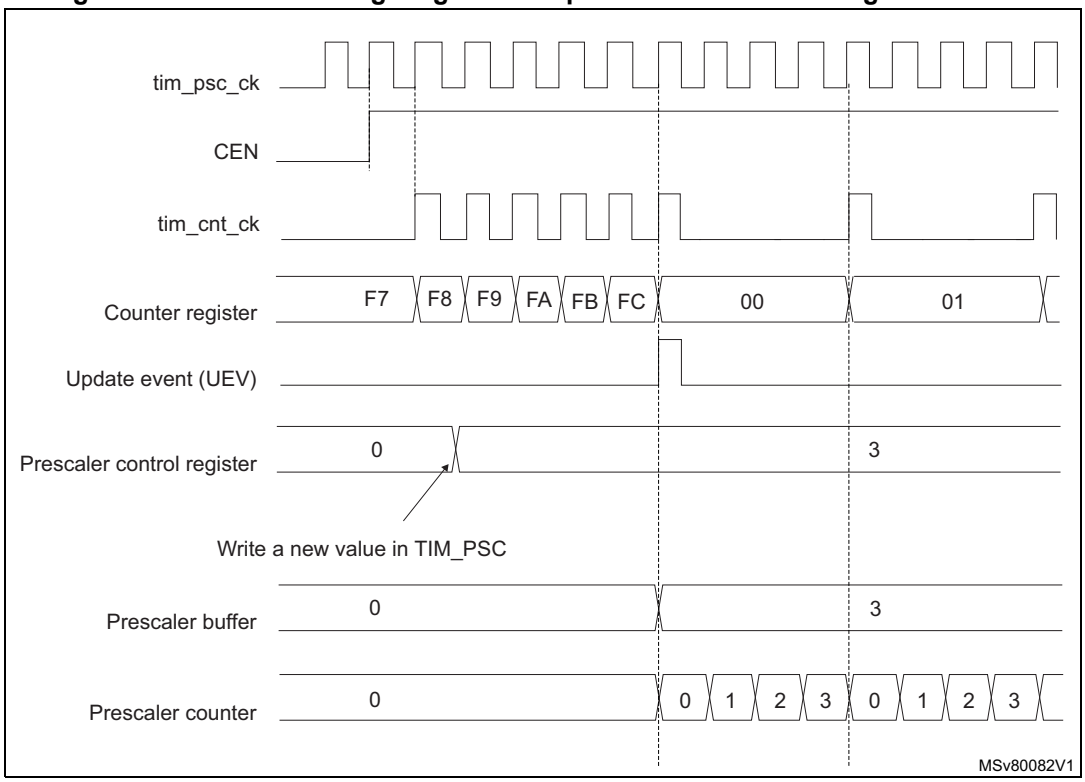


Figure 717. Counter timing diagram with prescaler division change from 1 to 4



36.3.5 Counting mode

The counter counts from 0 to the autoreload value (contents of the TIM_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIM_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIM_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIM_PSC register).
- The autoreload shadow register is updated with the preload value (TIM_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 718. Counter timing diagram, internal clock divided by 1

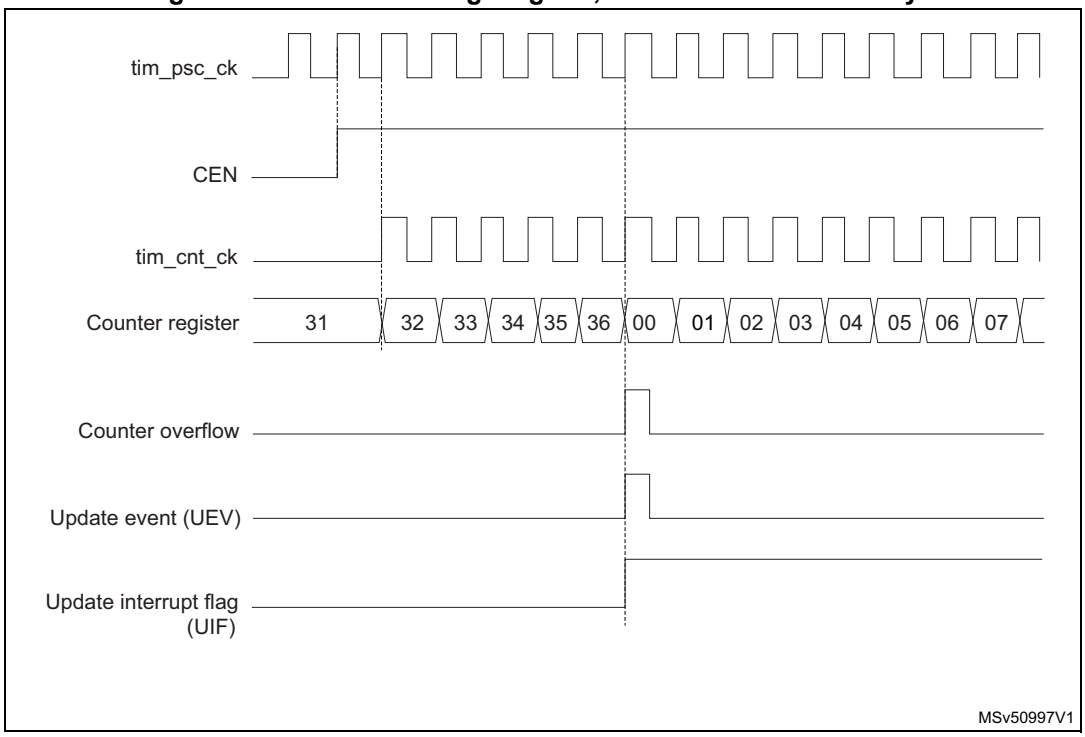


Figure 719. Counter timing diagram, internal clock divided by 2

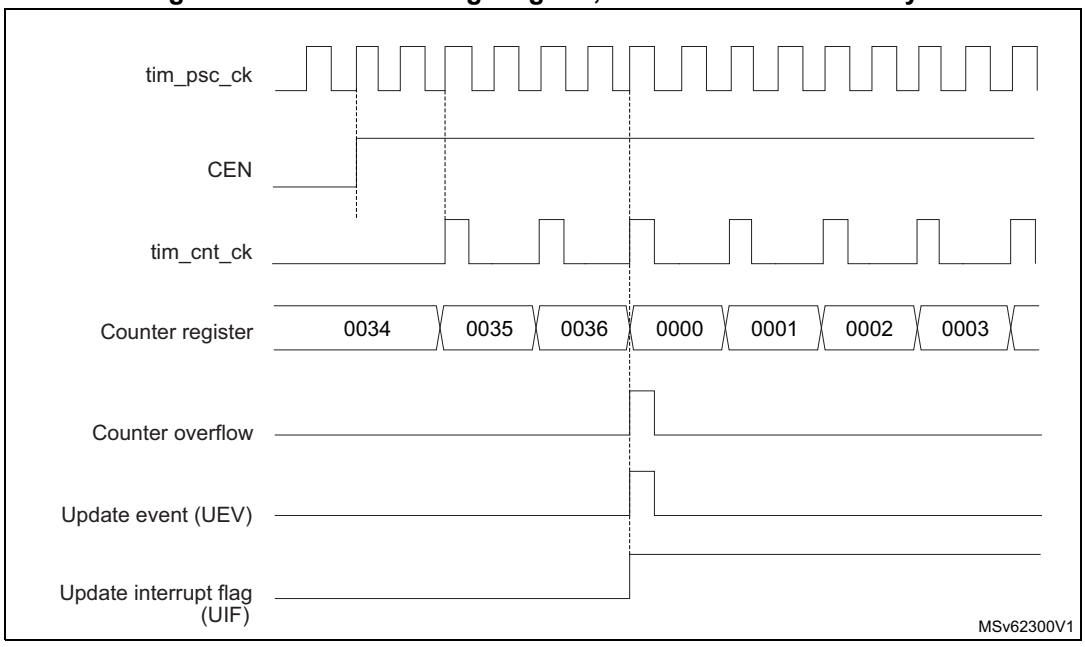


Figure 720. Counter timing diagram, internal clock divided by 4

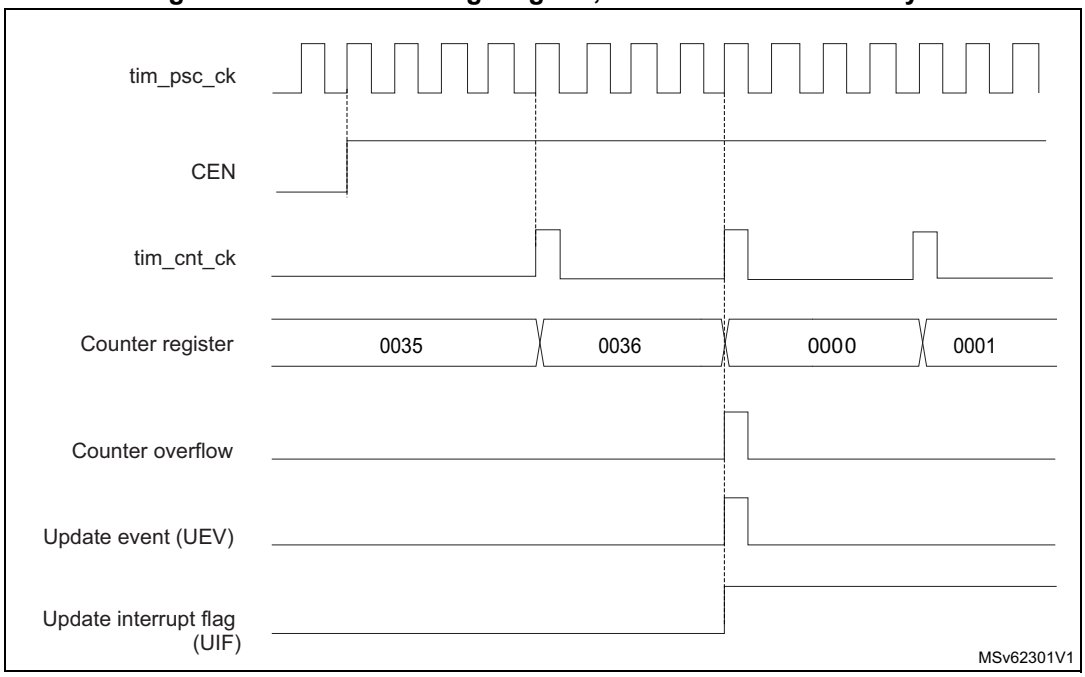


Figure 721. Counter timing diagram, internal clock divided by N

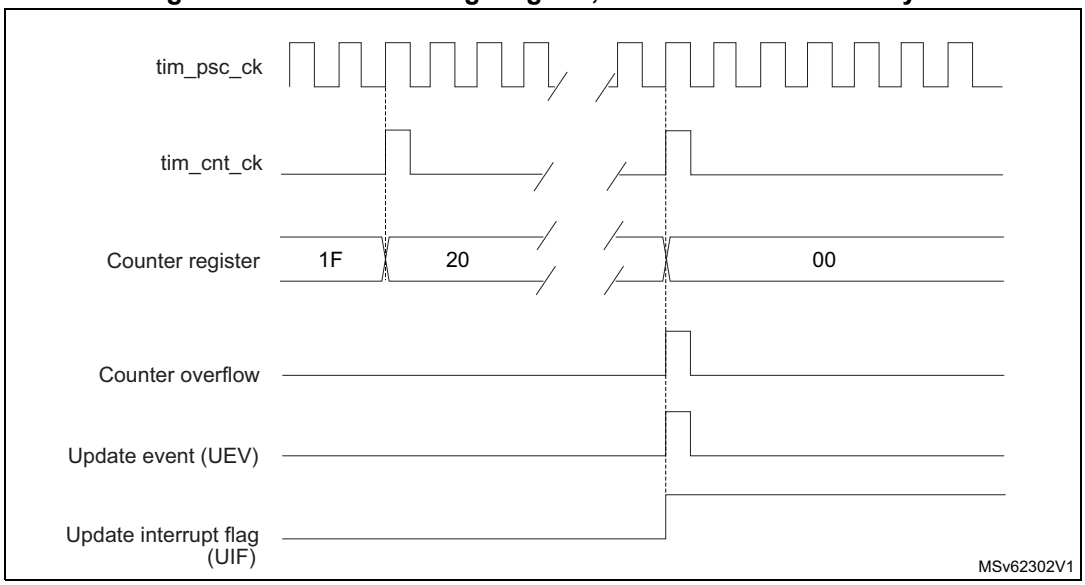


Figure 722. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded)

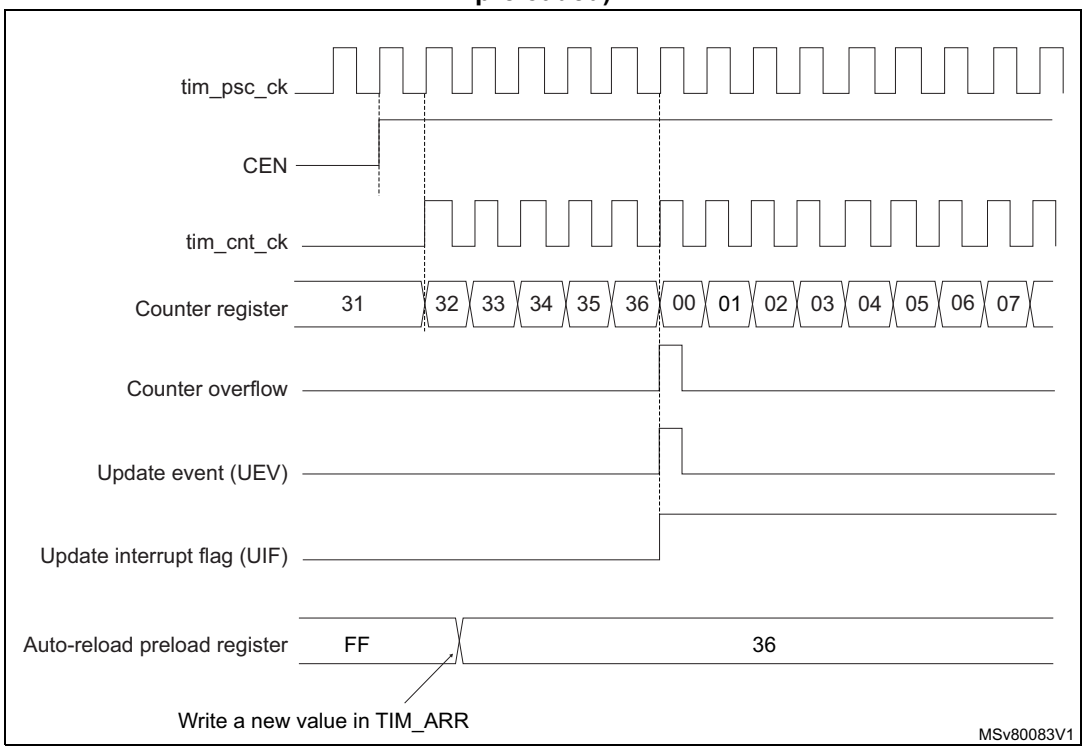
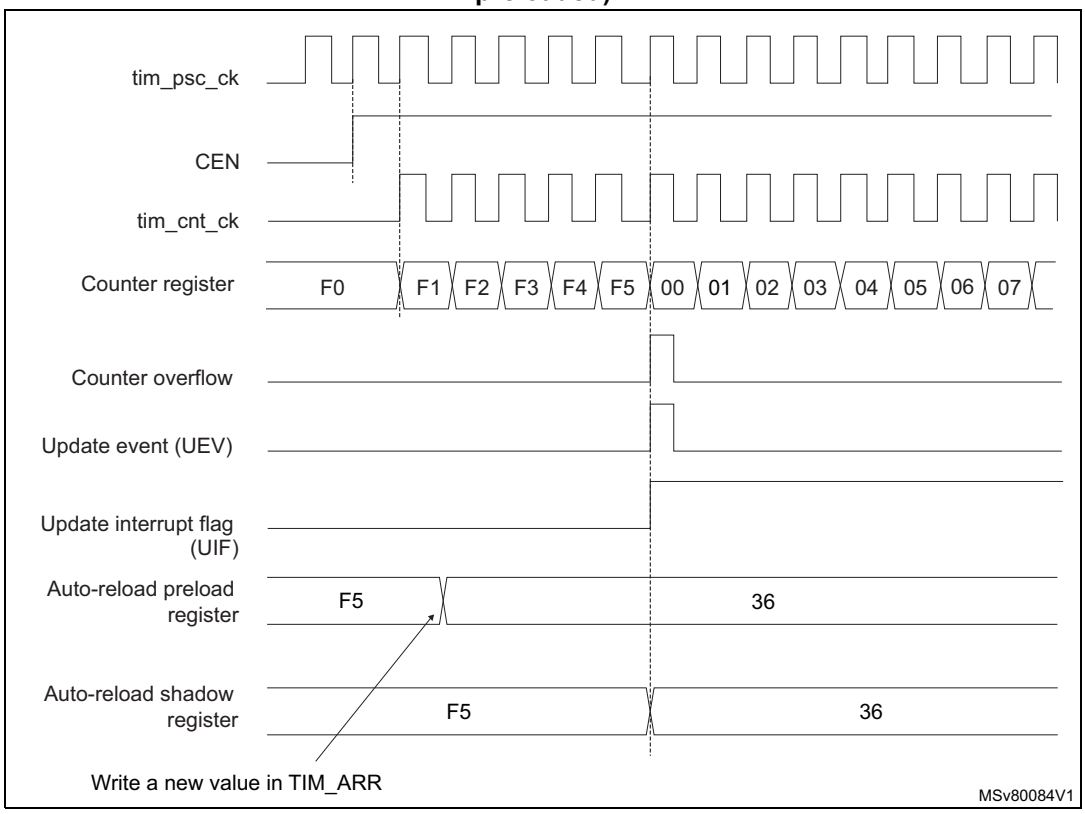


Figure 723. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded)



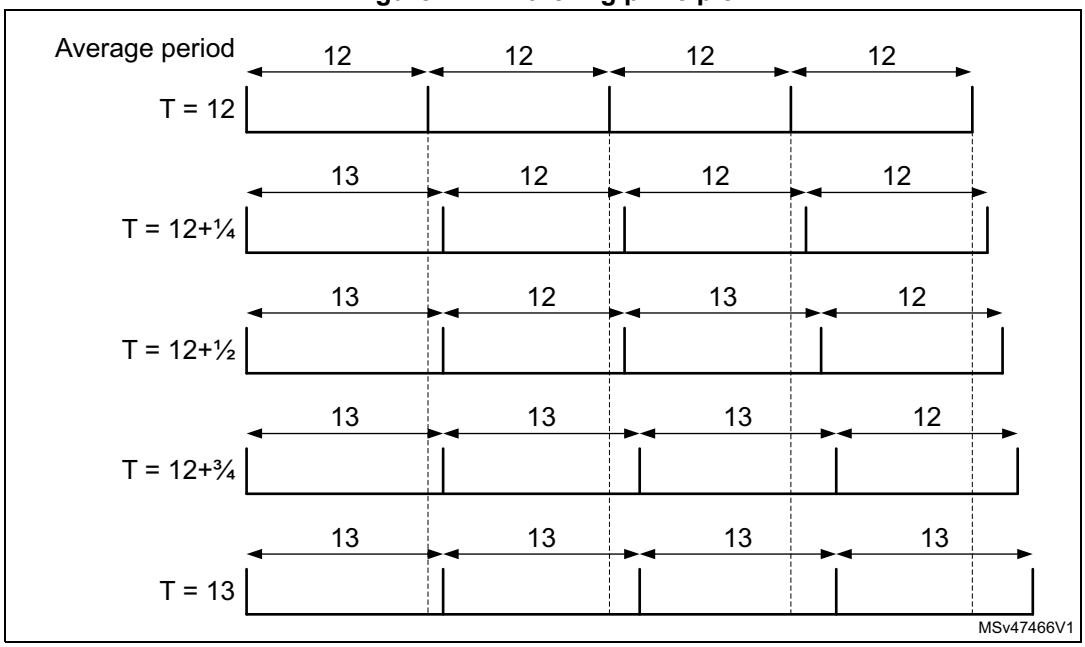
Dithering mode

The time base effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIM_CR1 register. This affects the way the TIM_ARR is behaving, and is useful for adjusting the average counter period when the timer is used as a trigger (typically for a DAC).

The operating principle is to have the actual ARR value slightly changed (adding or not one timer clock period) over 16 consecutive counting periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average counting period.

Figure 724 presents the dithering principle applied to four consecutive counting periods.

Figure 724. Dithering principle

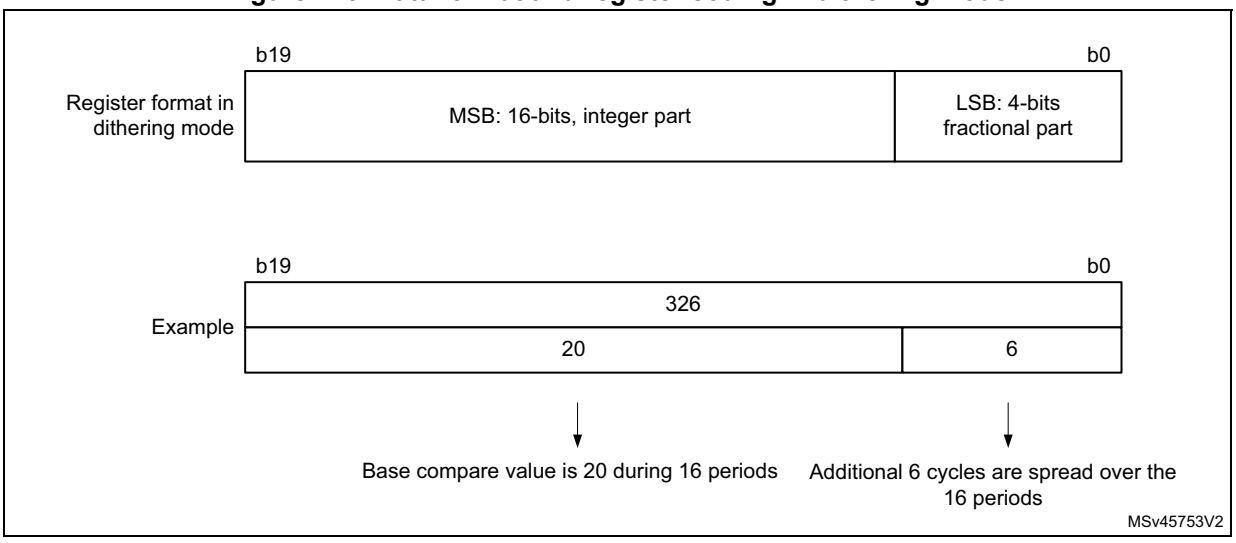


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 725* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The DITHEN bit must be reset
3. The CEN bit can be set (eventually with ARPE = 1).

Figure 725. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

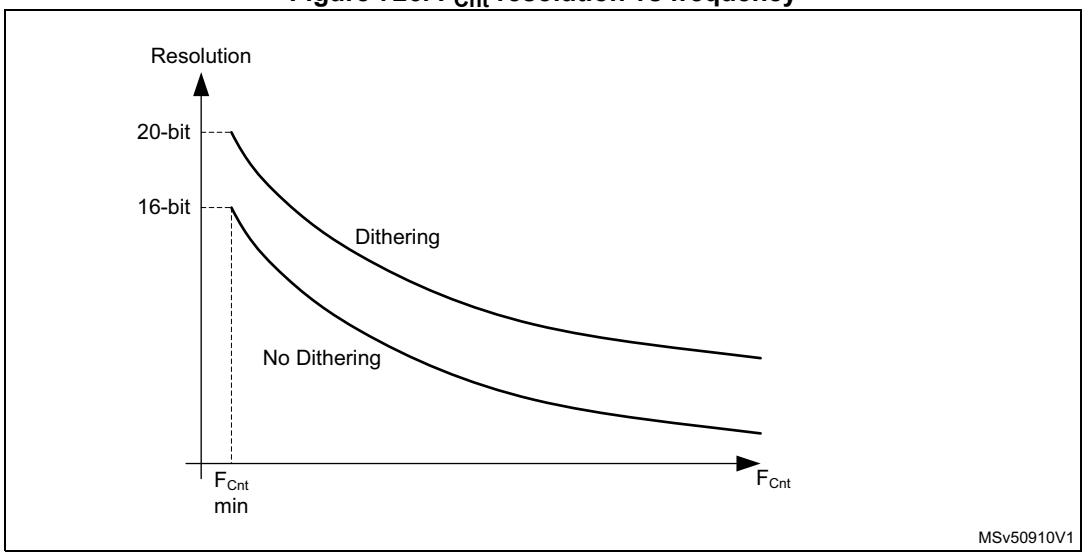
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIM_ARR value is limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

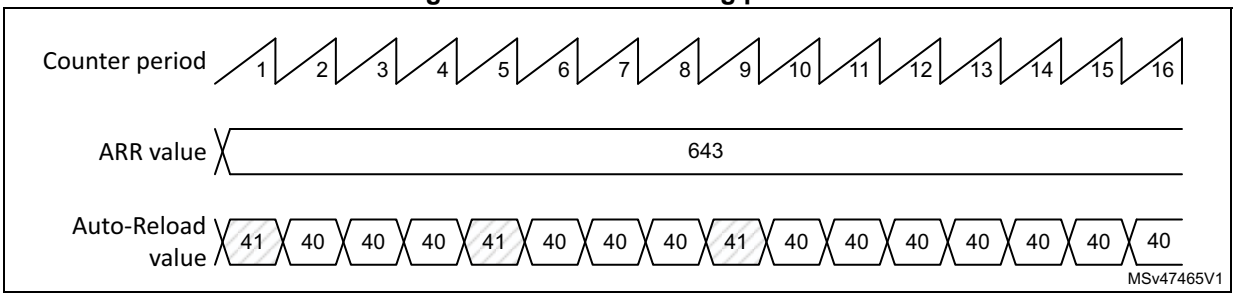
As shown on *Figure 726*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 726. F_{Cnt} resolution vs frequency



The period changes are spread over 16 consecutive periods, as described in *Figure 727*.

Figure 727. PWM dithering pattern



The autoreload and compare values increments are spread following the specific patterns described in *Table 414*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 414. TIM_ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

36.3.6 UIF bit remapping

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag UIF into the timer counter register's bit 31 (TIM_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

36.3.7 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receiving events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

36.3.8 ADC synchronization

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIM_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- When the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.
- When the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source).

1. The counter period must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times (\text{TIM_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times \text{TIM_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements are met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock
2. Configure the timer and set the ADSYNC bit
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits)
4. Start the timer (with the CEN counter enable bit)

36.3.9 TIM6/TIM7 DMA requests

The TIM6/TIM7 can generate a single DMA request, as shown in *Table 415*.

Table 415. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE

36.3.10 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter can either continue to work normally or be stopped.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

36.3.11 TIM6/TIM7 low-power modes

Table 416. Effect of low-power modes on TIM6/TIM7

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

36.3.12 TIM6/TIM7 interrupts

The TIM6/TIM7 can generate a single interrupt, as shown in *Table 417*.

Table 417. Interrupt request

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM6 TIM7	Update	UIF	UIE	write 0 in UIF	Yes	No

36.4 TIM6/TIM7 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

36.4.1 TIM control register 1 (TIM_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw				rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **ARPE**: Auto-reload preload enable
0: TIM_ARR register is not buffered.
1: TIM_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

CEN is cleared automatically in one-pulse mode, when an update event occurs.

36.4.2 TIM control register 2 (TIM_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 36.3.8* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bits 27:7 Reserved, must be kept at reset value.

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000:**Reset** - the UG bit from the TIM_EGR register is used as a trigger output (tim_trgo).

001:**Enable** - the Counter enable signal, tim_cnt_en, is used as a trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated when the CEN control bit is written.

010:**Update** - The update event is selected as a trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

Note: The clock of the slave timer or he peripheral receiving the tim_trgo must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bits 3:0 Reserved, must be kept at reset value.

36.4.3 TIM DMA/interrupt enable register (TIM_DIER)

Address offset: 0x0C

Reset value: 0x0000

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

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

36.4.4 TIM status register (TIM_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rc_w0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **UIOVRF**: Update Interrupt Overrun flag

This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.

0: No Update Interrupt Overrun

1: Update Interrupt Overrun

Bits 23:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

–On counter overflow if UDIS = 0 in the TIM_CR1 register.

–When CNT is reinitialized by software using the UG bit in the TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.

36.4.5 TIM event generation register (TIM_EGR)

Address offset: 0x14

Reset value: 0x0000

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

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

36.4.6 TIM counter (TIM_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIM_ISR register. If the UIFREMAP bit in TIM_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

36.4.7 TIM prescaler (TIM_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency $f_{tim_cnt_ck}$ is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIM_EGR register.

36.4.8 TIM autoreload register (TIM_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.

Refer to *Section 36.3.4: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

36.4.9 TIM6/TIM7 register map

TIM6/TIM7 registers are mapped as 16-bit addressable registers as described in the table below:

Table 418. TIM6/TIM7 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIM_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																				0	0				0				0	0	0	0
0x04	**TIM_CR2**	Res.	Res.	Res.	ADSYNC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS [2:0]			Res.	Res.	Res.	Res.
	Reset value				0																						0	0	0				
0x08	Reserved	Res.																															
0x0C	**TIM_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value																								0								0
0x10	**TIM_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
	Reset value								0																								0
0x14	**TIM_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
	Reset value																																0
0x18-0x20	Reserved	Res.																															
0x24	**TIM_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

37 General-purpose timers (TIM9/TIM10/TIM11/TIM12/TIM13/TIM14)

37.1 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 introduction

The TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 general-purpose timers consist in a 16-bit autoreload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 37.4.21: Timer synchronization (TIM9/TIM12 only)*.

37.2 TIM9/TIM12 main features

The features of the TIM9/TIM12 general-purpose timers include:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Up to two independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal trigger)
  - Trigger event (counter start, stop, initialization, or count by internal trigger)
  - Input capture
  - Output compare
- ADC synchronization for jitter-free sampling points

37.3 TIM10/TIM11/TIM13/TIM14 main features

The features of general-purpose timers TIM10/TIM11/TIM13/TIM14 include:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Independent channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare

37.4 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 functional description

37.4.1 Block diagram

Figure 728. General-purpose timer block diagram (TIM9/TIM12)

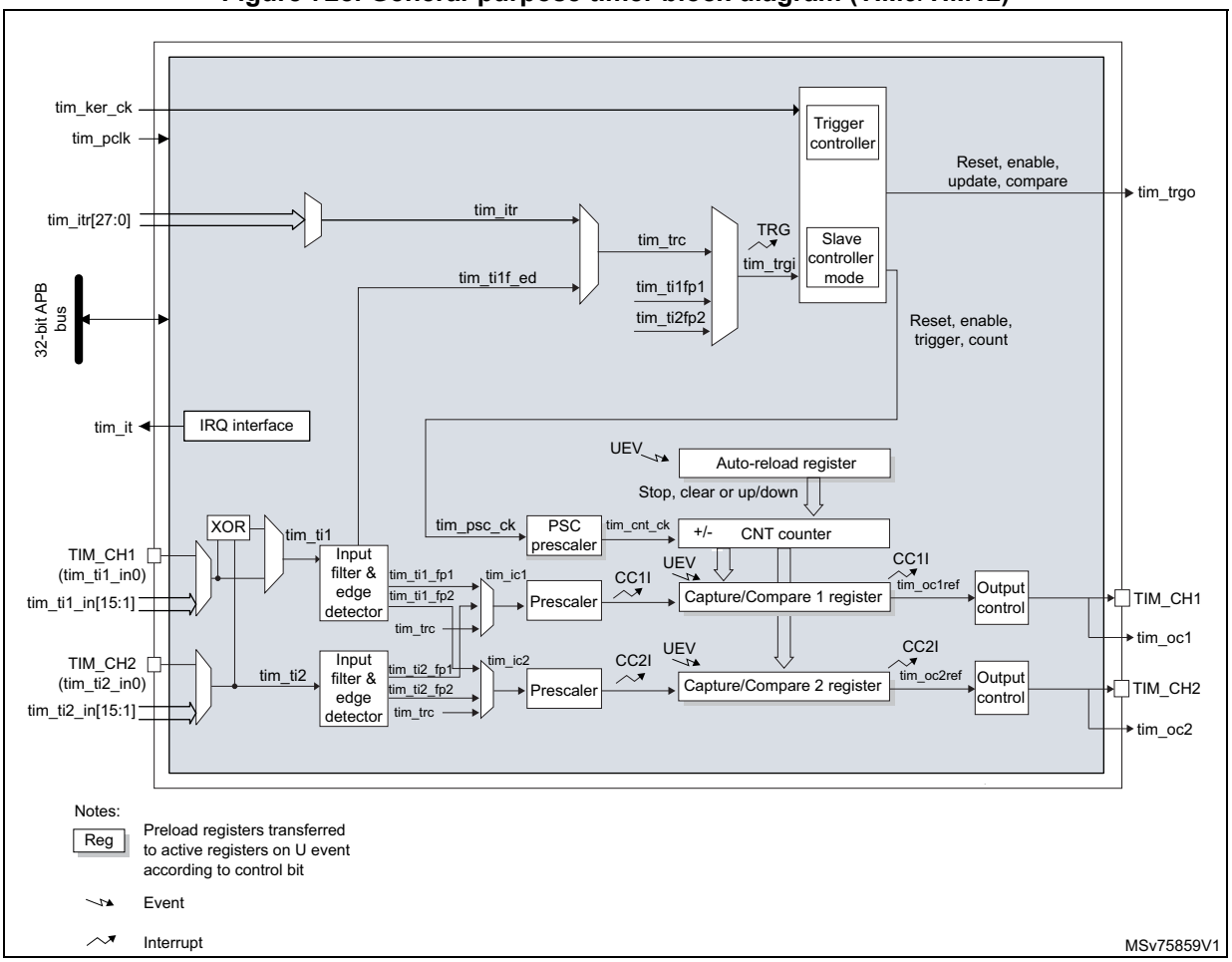
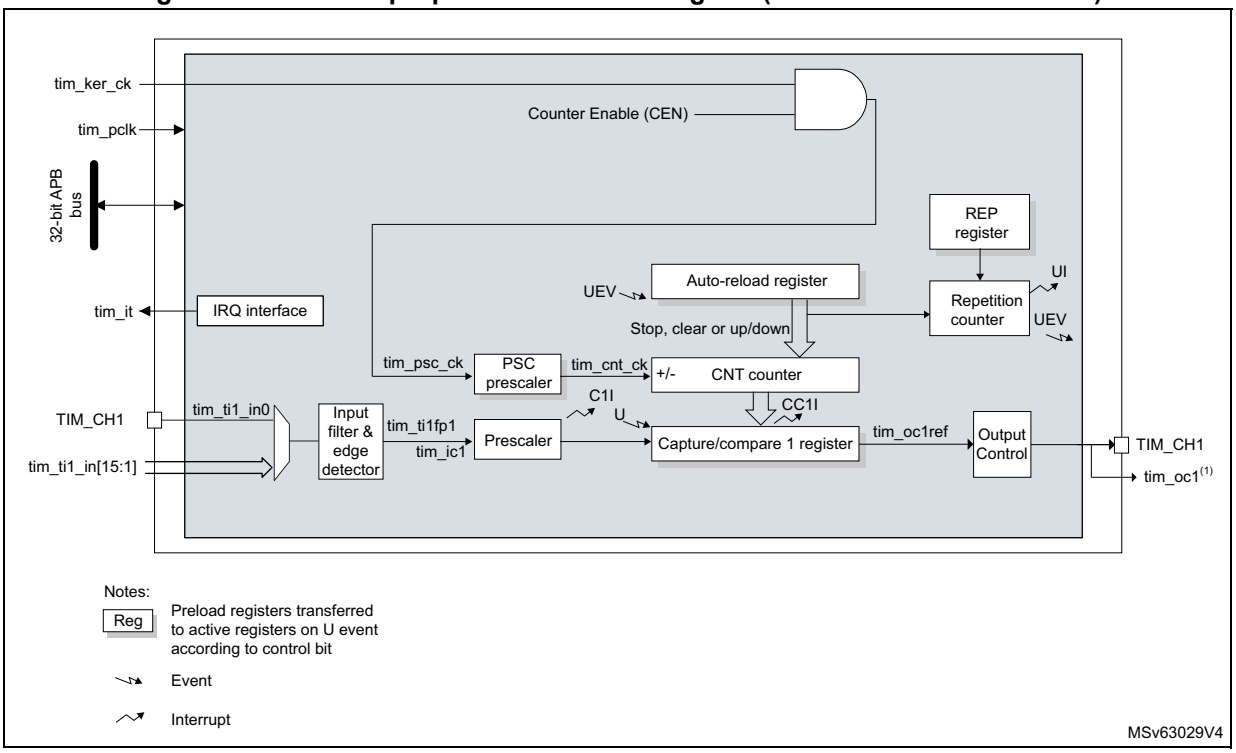


Figure 729. General-purpose timer block diagram (TIM10/TIM11/TIM13/TIM14)



1. This signal can be used as trigger for some slave timer (see internal trigger connection table in next section). See *Section 37.4.22: Using timer output as trigger for other timers (TIM10/TIM11/TIM13/TIM14 only)* for details.

37.4.2 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 pins and internal signals

Table 419 and *Table 420* in this section summarize the TIM inputs and outputs.

Table 419. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2(1)	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger inputs.

1. Available for TIM9/TIM12 only.

Table 420. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0](1)	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock).
tim_itr[27:0](1)	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_oc1 tim_oc2(1)	Output	Internal timer output. Can be used for triggering other timers or the ADC(s).

Table 420. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_trgo(1)	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, and trigger requests

1. Available for TIM9/TIM12 only.

Table 421 and *Table 422* list the sources connected to the tim_ti[2:1] input multiplexers.

Table 421. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources					
	TIM9	TIM10	TIM11	TIM12	TIM13	TIM14
tim_ti1_in0	TIM9_CH1	TIM10_CH1	TIM11_CH1	TIM12_CH1	TIM13_CH1	TIM14_CH1
tim_ti1_in1	comp1_out	comp3_out	comp4_out	comp5_out	comp7_out	comp8_out
tim_ti1_in2	rcc_mco1	i3c1_ibi_ack	Reserved	rcc_mco1	i3c1_ibi_ack	Reserved
tim_ti1_in3	rcc_mco2	Reserved		rcc_mco2	Reserved	
tim_ti1_in4	hsi_cal_ck			hsi_cal_ck		
tim_ti1_in5	Reserved			Reserved		
tim_ti1_in6	usb_sof			usb_sof		
tim_ti1_in7	tim1_trgo			tim4_trgo		
tim_ti1_in8	tim2_trgo			tim20_trgo		
tim_ti1_in[15:9]	Reserved					

Table 422. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources	
	TIM9	TIM12
tim_ti2_in0	TIM9_CH2	TIM12_CH2
tim_ti2_in1	comp2_out	comp6_out
tim_ti2_in2	Reserved	Reserved
tim_ti2_in3		
tim_ti2_in4		
tim_ti2_in5		
tim_ti2_in6	tim15_trgo	tim15_trgo

Table 422. Interconnect to the tim_ti2 input multiplexer (continued)

tim_ti2 inputs	Sources	
	TIM9	TIM12
tim_ti2_in7	tim3_trgo	tim5_trgo
tim_ti2_in8	tim8_trgo	Reserved
tim_ti2_in[15:9]	Reserved	

Table 423 lists the internal sources connected to the tim_itr input multiplexer.

Table 423. TIMx internal trigger connection

tim_itrx inputs	TIM9	TIM12
tim_itr0	tim1_trgo	tim1_trgo
tim_itr1	tim1_trgo2	tim1_trgo2
tim_itr2	tim2_trgo	tim2_trgo
tim_itr3	tim3_trgo	tim3_trgo
tim_itr4	tim4_trgo	tim4_trgo
tim_itr5	tim5_trgo	tim5_trgo
tim_itr6	tim8_trgo	tim8_trgo
tim_itr7	tim8_trgo2	tim8_trgo2
tim_itr8	Reserved	tim9_trgo
tim_itr9	tim10_oc1	tim10_oc1
tim_itr10	tim11_oc1	tim11_oc1
tim_itr11	tim12_trgo	Reserved
tim_itr12	tim13_oc1	tim13_oc1
tim_itr13	tim14_oc1	tim14_oc1
tim_itr14	tim15_trgo	tim15_trgo
tim_itr15	tim16_oc1	tim16_oc1
tim_itr16	tim17_oc1	tim17_oc1
tim_itr17	tim20_trgo	tim20_trgo
tim_itr18	tim20_trgo2	tim20_trgo2
tim_itr19	Reserved	Reserved
tim_itr20		
tim_itr21	hrtim1_sync_out2	hrtim1_sync_out2
tim_itr22	hrtim2_sync_out2	hrtim2_sync_out2
tim_itr[27:23]	Reserved	Reserved

37.4.3 Time-base unit

The main block of the timer is a 16-bit up-counter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIM_CNT)
- Prescaler register (TIM_PSC)
- Autoreload register (TIM_ARR).

The autoreload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIM_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIM_CR1 register. It can also be generated by software. The generation of the update event is described in details for each configuration.

Note: An update overrun status bit (UIOVRF in TIM_SR register) indicates that an update request occurred while an update interrupt is pending (update interrupt flag set). This bit is useful when debugging closed loop applications, to indicate that:

- *The update interrupt service routine is too long with respect to the interval between consecutive update events.*
- *The update interrupt service routine was served too late with respect to the subsequent update event.*

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIM_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting one clock cycle after setting the CEN bit in the TIM_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIM_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 730 and *Figure 731* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 730. Counter timing diagram with prescaler division change from 1 to 2

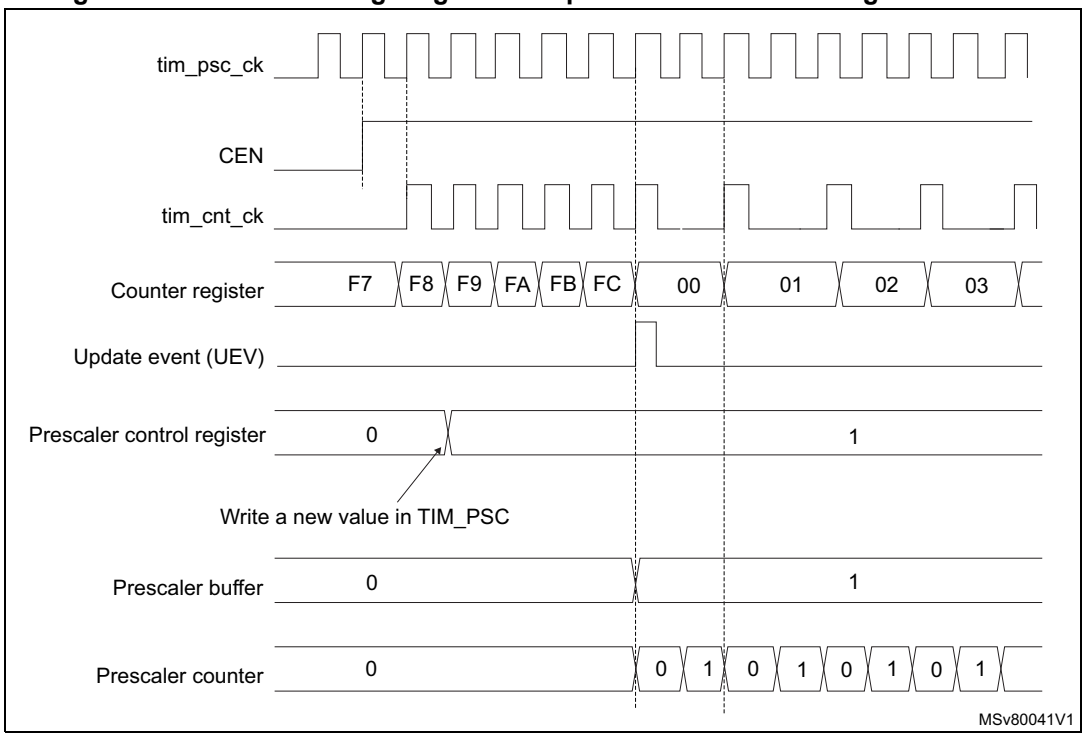
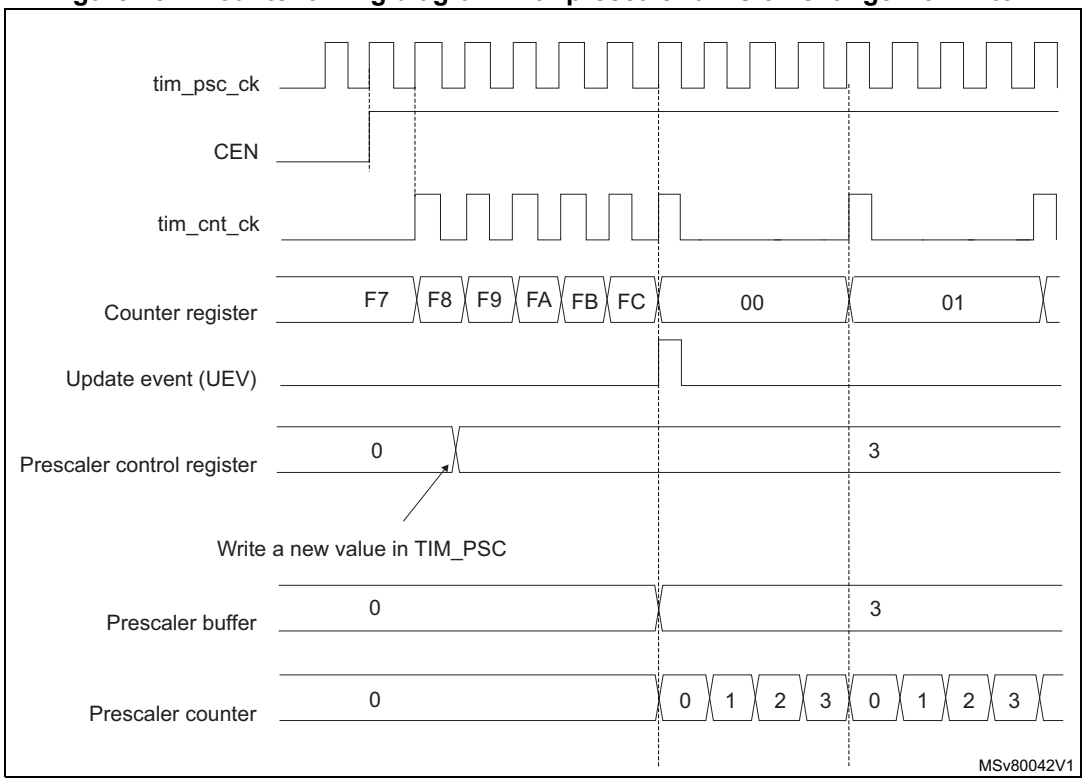


Figure 731. Counter timing diagram with prescaler division change from 1 to 4



37.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller on TIM9/TIM12) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The autoreload shadow register is updated with the preload value (TIM_ARR).
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 732. Counter timing diagram, internal clock divided by 1

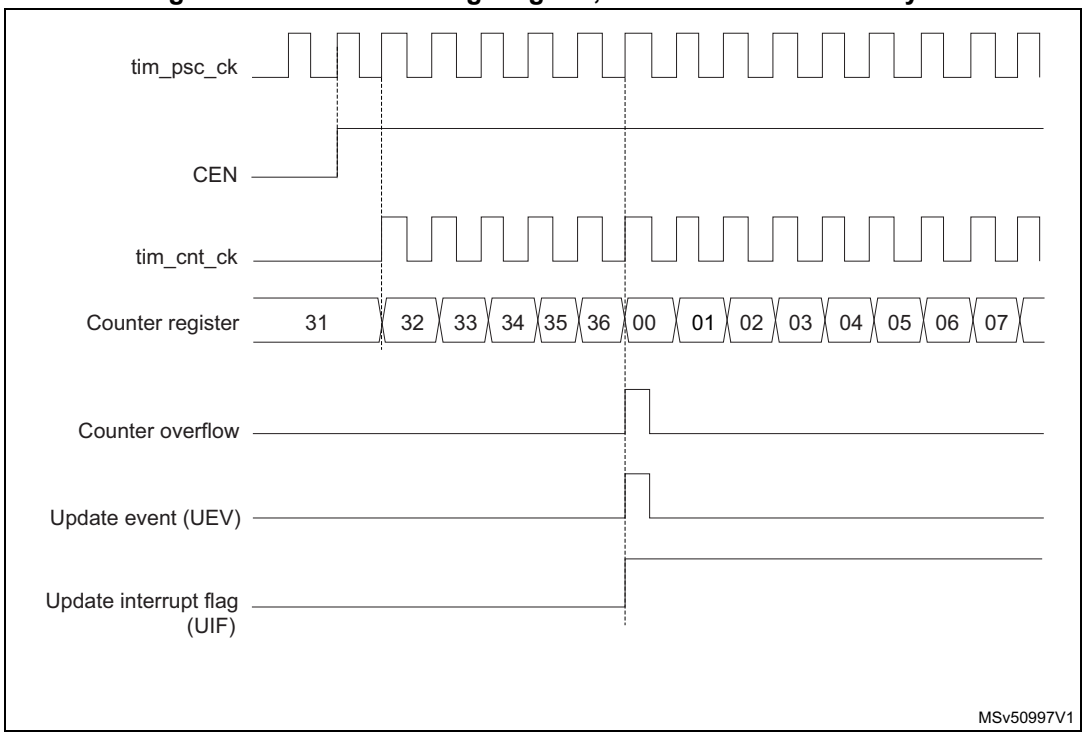


Figure 733. Counter timing diagram, internal clock divided by 2

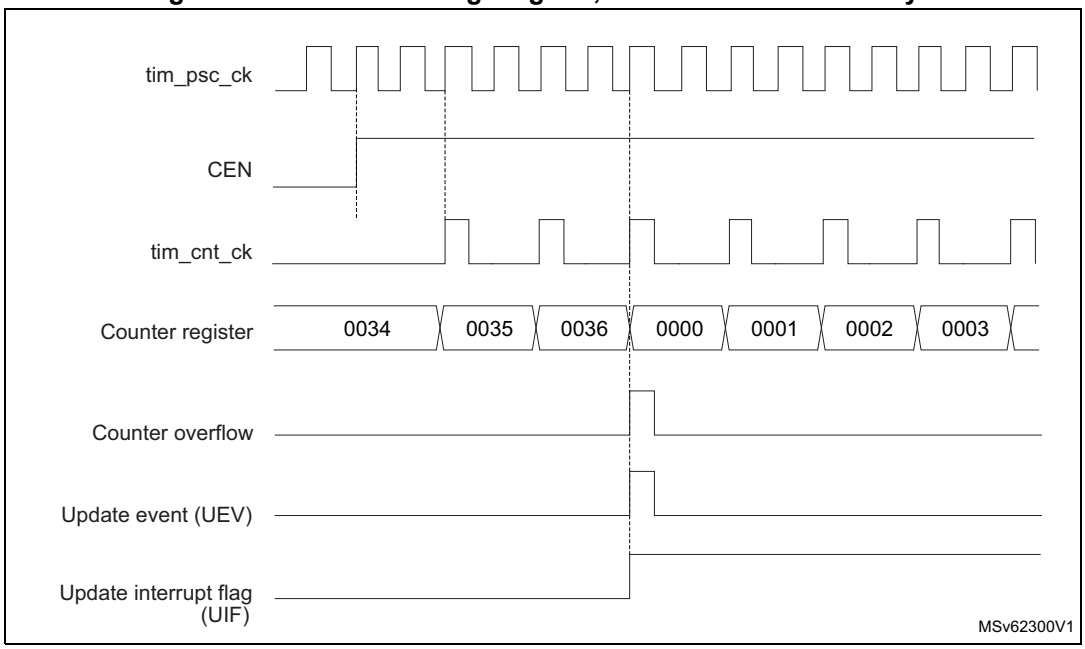


Figure 734. Counter timing diagram, internal clock divided by 4

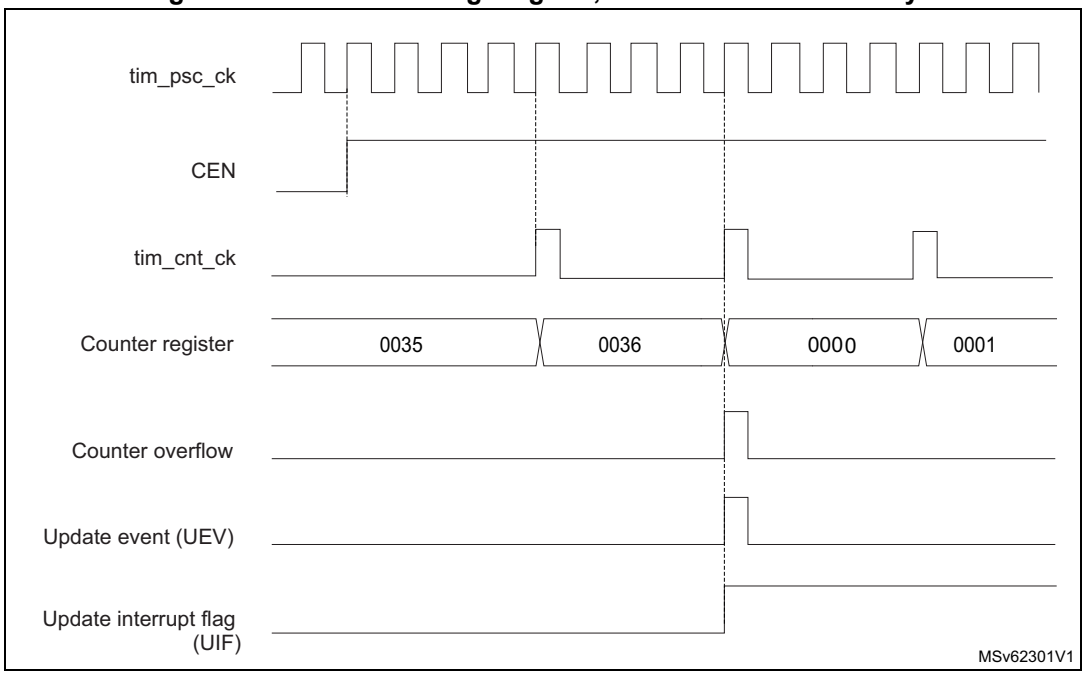


Figure 735. Counter timing diagram, internal clock divided by N

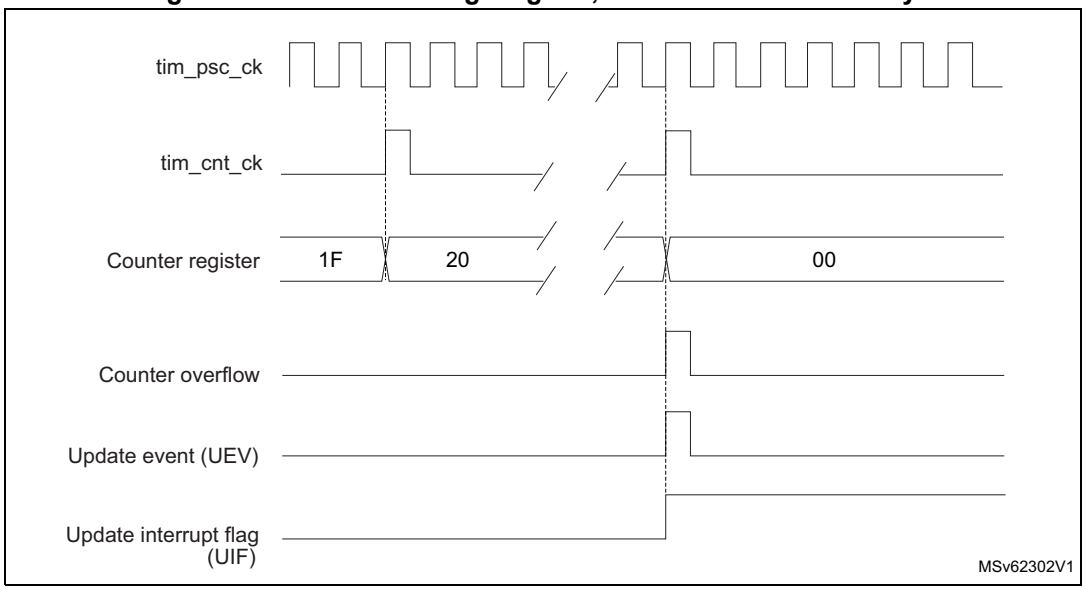


Figure 736. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded)

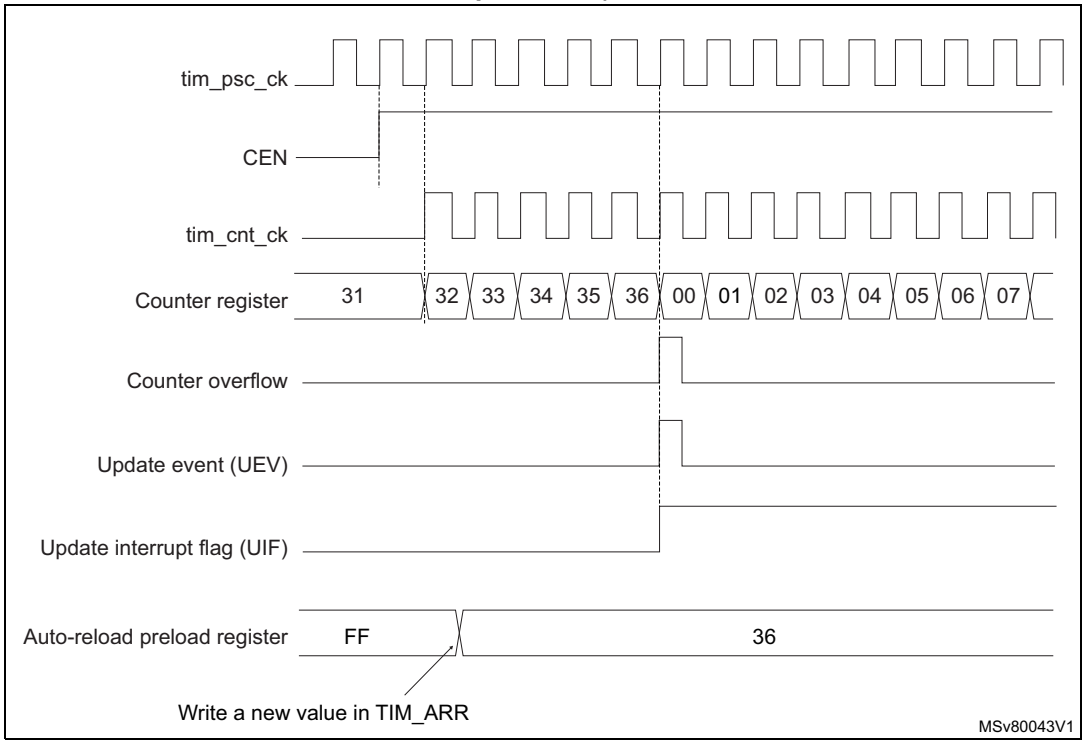
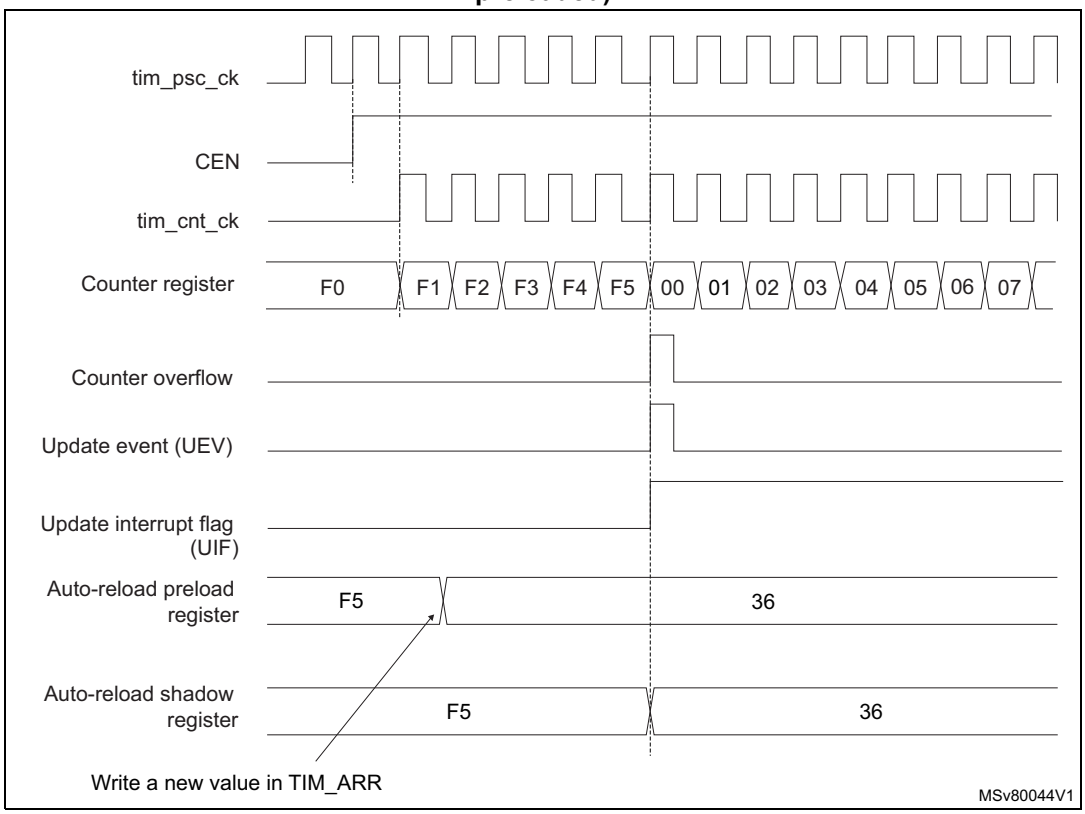


Figure 737. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded)



37.4.5 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (tim_ker_ck)
- External clock mode1 (for TIM9/TIM12): external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx) (for TIM9/TIM12): connecting the trigger output from another timer. For instance, another timer can be configured as a prescaler for TIM12. Refer to *Section : Using one timer as prescaler for another timer* for more details.

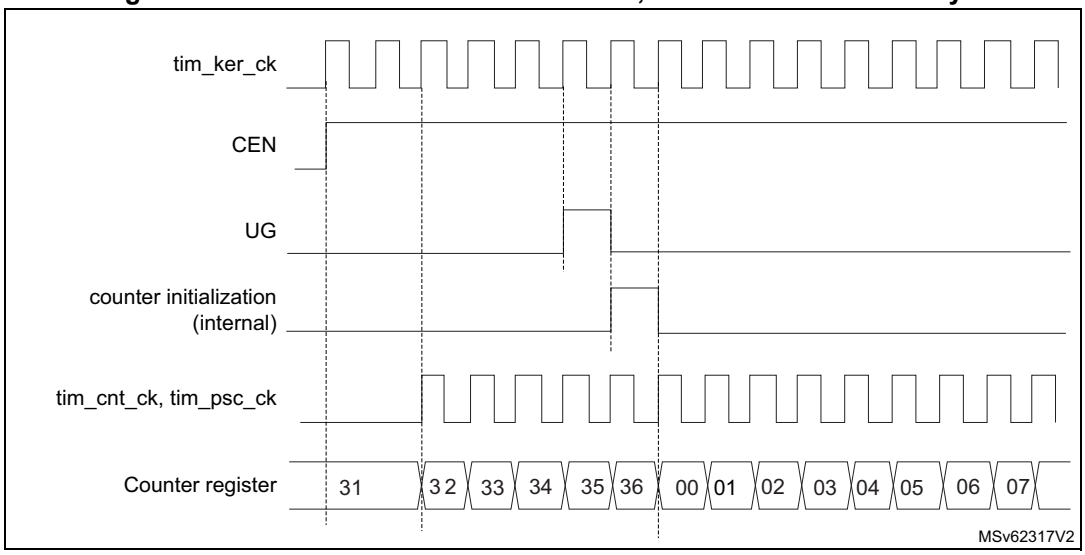
Internal clock source (tim_ker_ck)

The internal clock source is the default clock source for TIM10/TIM11/TIM13/TIM14.

For TIM9/TIM12, the internal clock source is selected when the slave mode controller is disabled (SMS = 000). The CEN bit in the TIM_CR1 register and the UG bit in the TIM_EGR register are then used as control bits and can be changed only by software (except for UG which remains cleared). As soon as the CEN bit is programmed to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 738 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

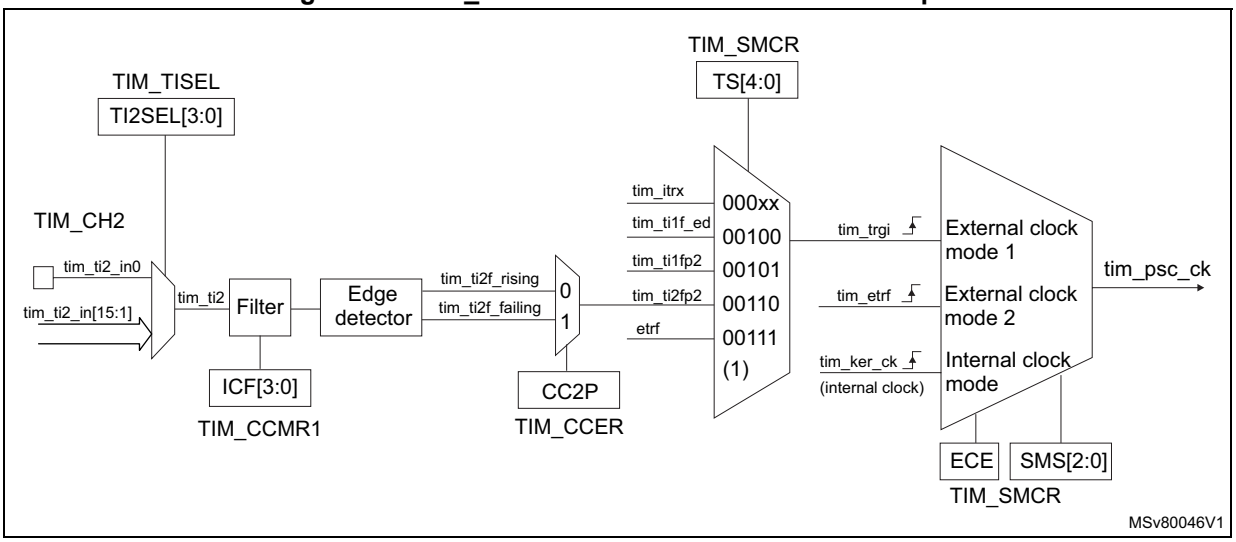
Figure 738. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1 (TIM9/TIM12 only)

This mode is selected when SMS = 111 in the TIM_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 739. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

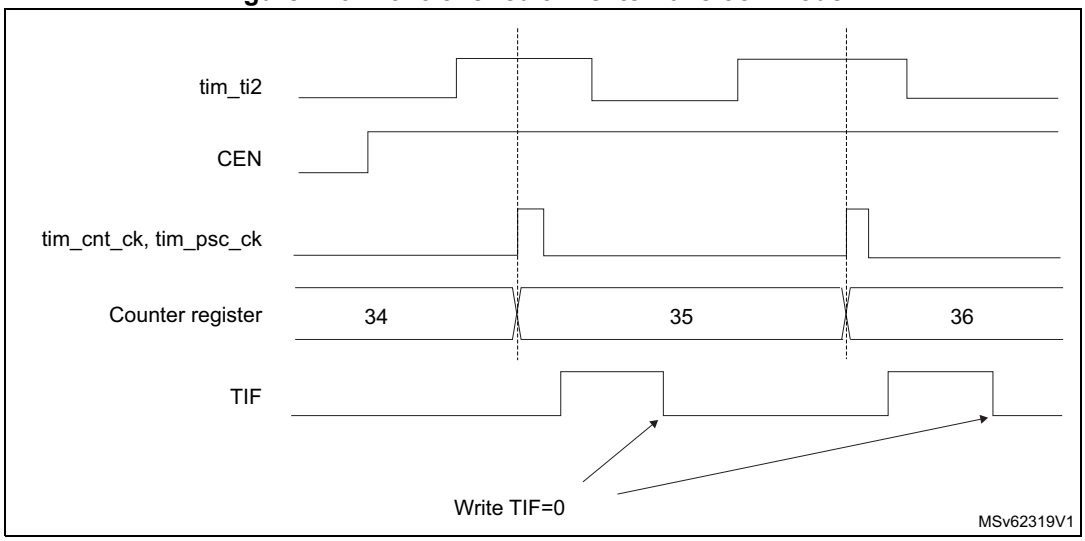
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIM_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIM_CCMR1 register (if no filter is needed, keep IC2F = 0000).
4. Select the rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIM_SMCR register.
6. Select tim_ti2 as the trigger input source by writing TS = 110 in the TIM_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 740. Control circuit in external clock mode 1



37.4.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler), and an output stage (with comparator and output control).

Figure 741, *Figure 742*, *Figure 743* and *Figure 744* give an overview of a capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 741. Capture/compare channel 1 input stage (TIM10/TIM11/TIM13/TIM14)

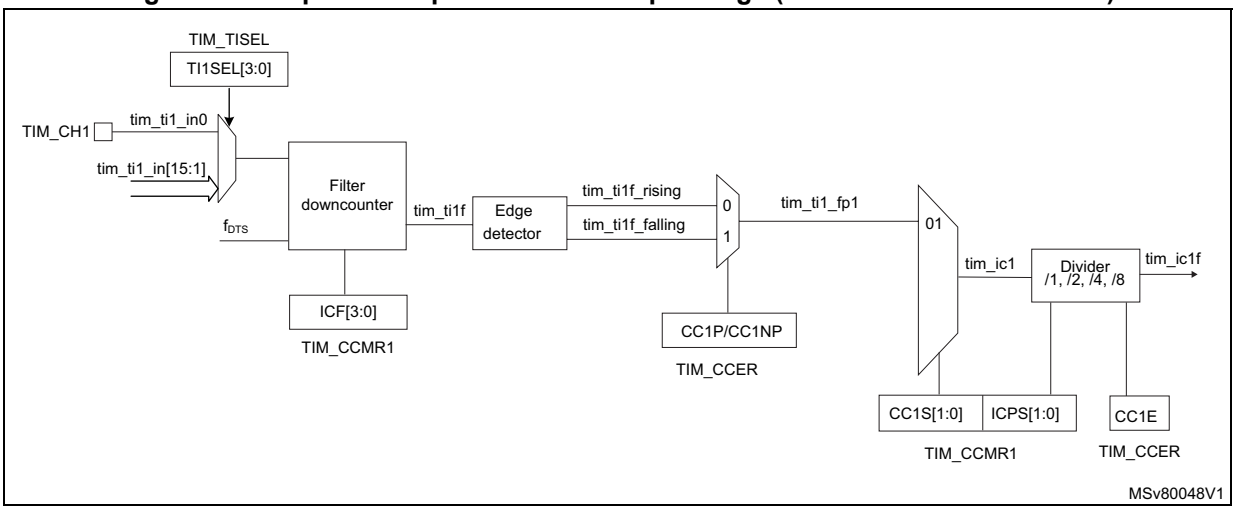
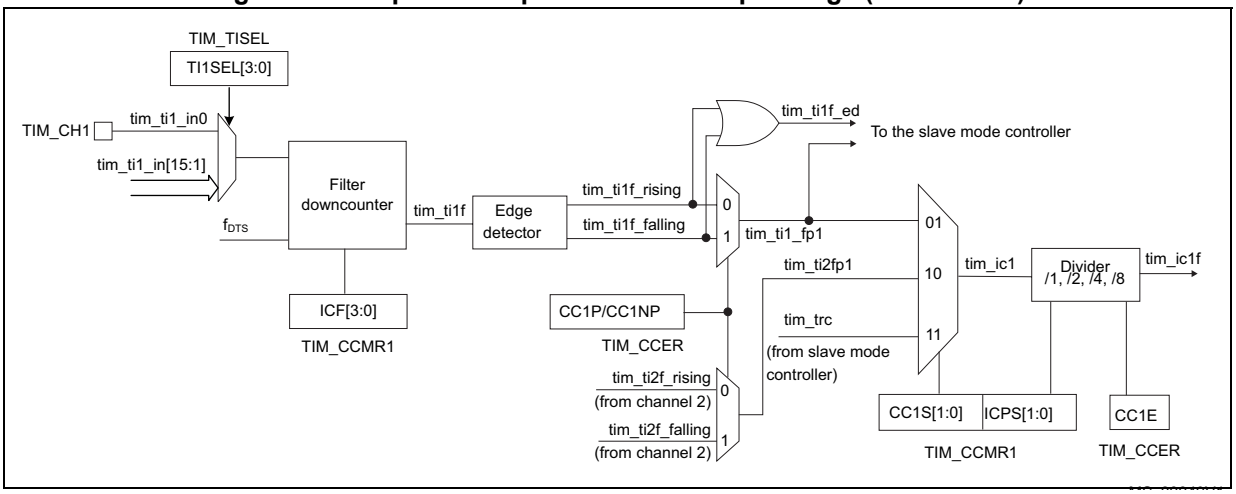


Figure 742. Capture/compare channel 1 input stage (TIM9/TIM12)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 743. Capture/compare channel 1 main circuit

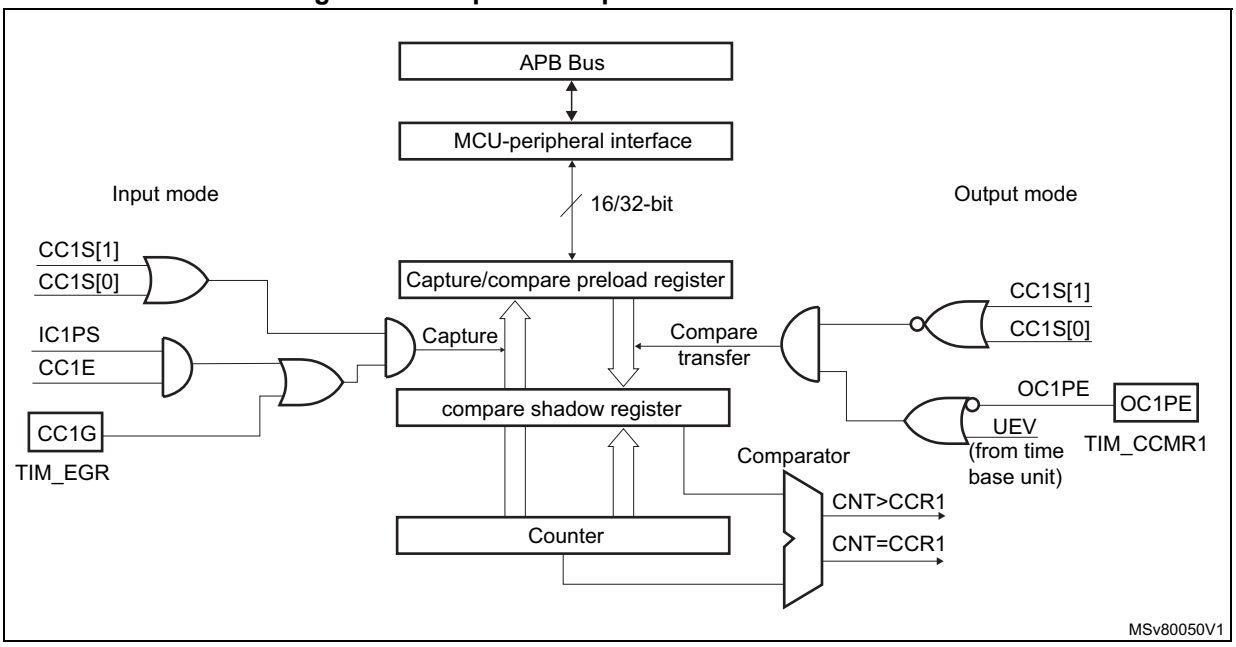
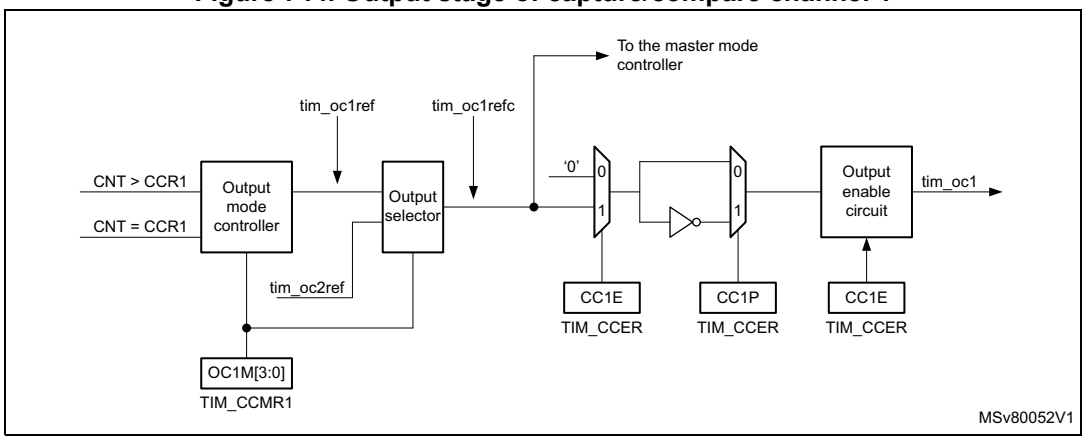


Figure 744. Output stage of capture/compare channel 1



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

37.4.7 Input capture mode

In Input capture mode, the capture/compare registers (TIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIM_SR register) is set and an interrupt request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIM_SR register) is set. CCxIF can be

cleared by software by writing it to 0 or by reading the captured data stored in the TIM_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIM_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input: TIM_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIM_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input mode, and the TIM_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (by programming the ICxF bits in the TIM_CCMRx register if the input is one of the tim_tix inputs). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. The user must program a filter duration longer than these five clock cycles. The user can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIM_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by programming CC1P and CC1NP bits to 00 in the TIM_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the user wishes the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIM_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIM_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIM_DIER register.

When an input capture occurs:

- The TIM_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt requests can be generated by software by setting the corresponding CCxG bit in the TIM_EGR register.

37.4.8 PWM input mode (TIM9/TIM12 only)

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIM_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges.

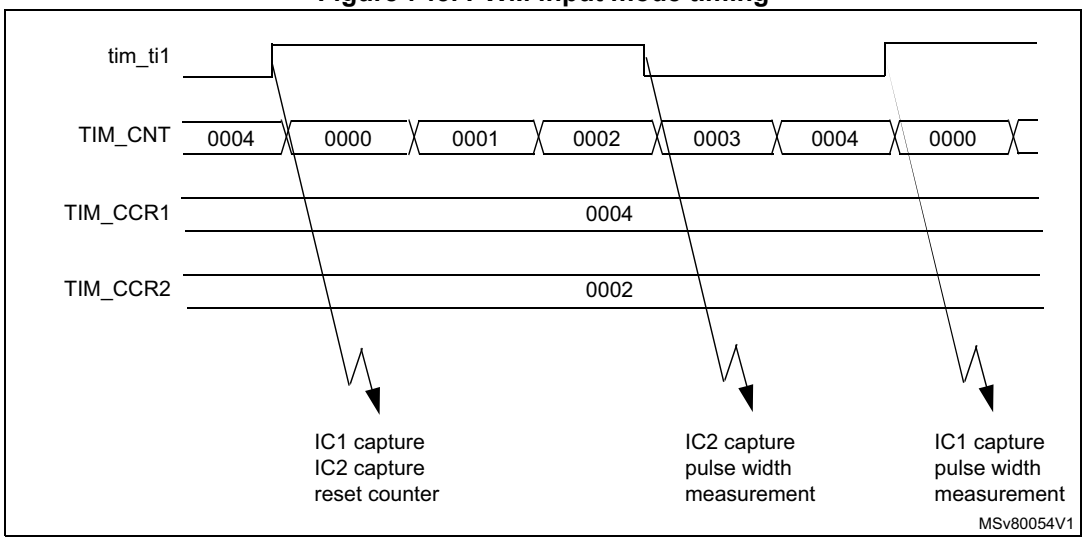
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These two tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIM_CCR1 register) and the duty cycle (in TIM_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input for TIM_CCR1: write the CC1S bits to 01 in the TIM_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIM_CCR1 and counter clear): program the CC1P and CC1NP bits to 00 (active on rising edge).
4. Select the active input for TIM_CCR2: write the CC2S bits to 10 in the TIM_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIM_CCR2): program the CC2P and CC2NP bits to 10 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIM_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIM_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIM_CCER register.

Figure 745. PWM input mode timing



37.4.9 Forced output mode

In output mode (CCxS bits = 00 in the TIM_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 0101 in the OCxM bits in the corresponding TIM_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 0100 in the TIM_CCMRx register.

Anyway, the comparison between the TIM_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

37.4.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

1. Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIM_CCMRx register) and the output polarity (CCxP bit in the TIM_CCER register). The output pin can keep its level (OCxM = 0000), be set active (OCxM = 0001), be set inactive (OCxM = 0010) or can toggle (OCxM = 0011) on match.
2. Sets a flag in the interrupt status register (CCxIF bit in the TIM_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCxIE bit in the TIM_DIER register).

The TIM_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIM_CCMRx register.

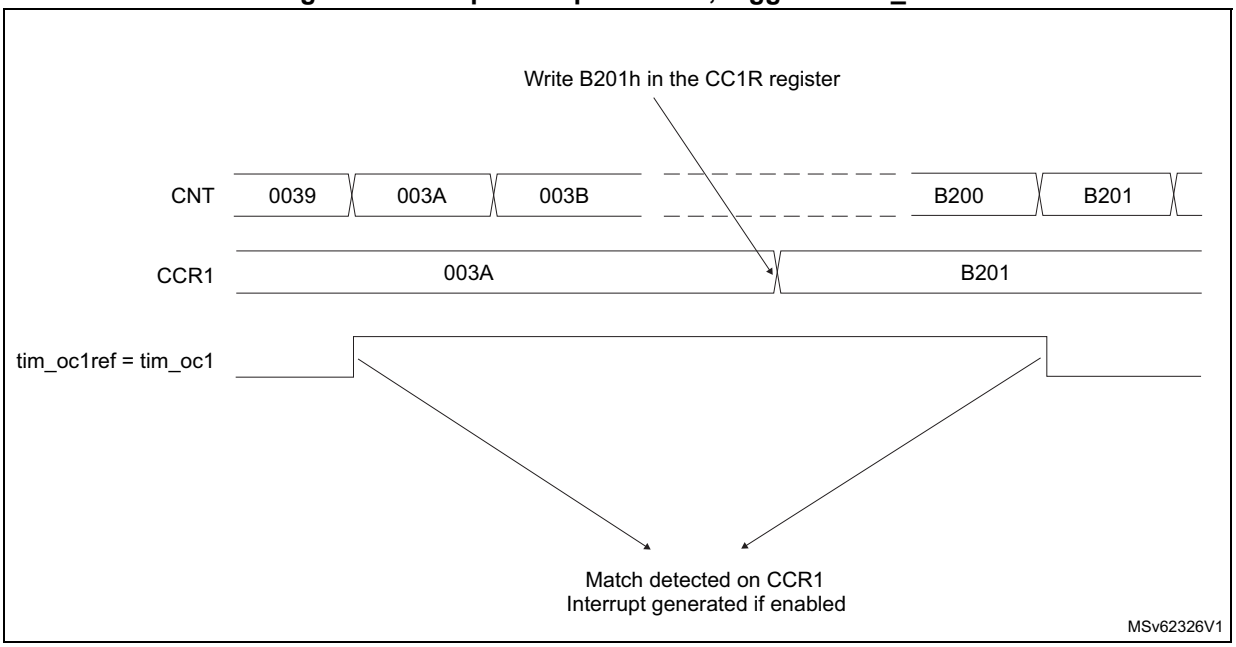
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIM_ARR and TIM_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIM_CR1 register.

The TIM_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIM_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 746*.

Figure 746. Output compare mode, toggle on tim_oc1.



37.4.11 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIM_ARR register and a duty cycle determined by the value of the TIM_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 0110 (PWM mode 1) or 0111 (PWM mode 2) in the OCxM bits in the TIM_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIM_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIM_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIM_EGR register.

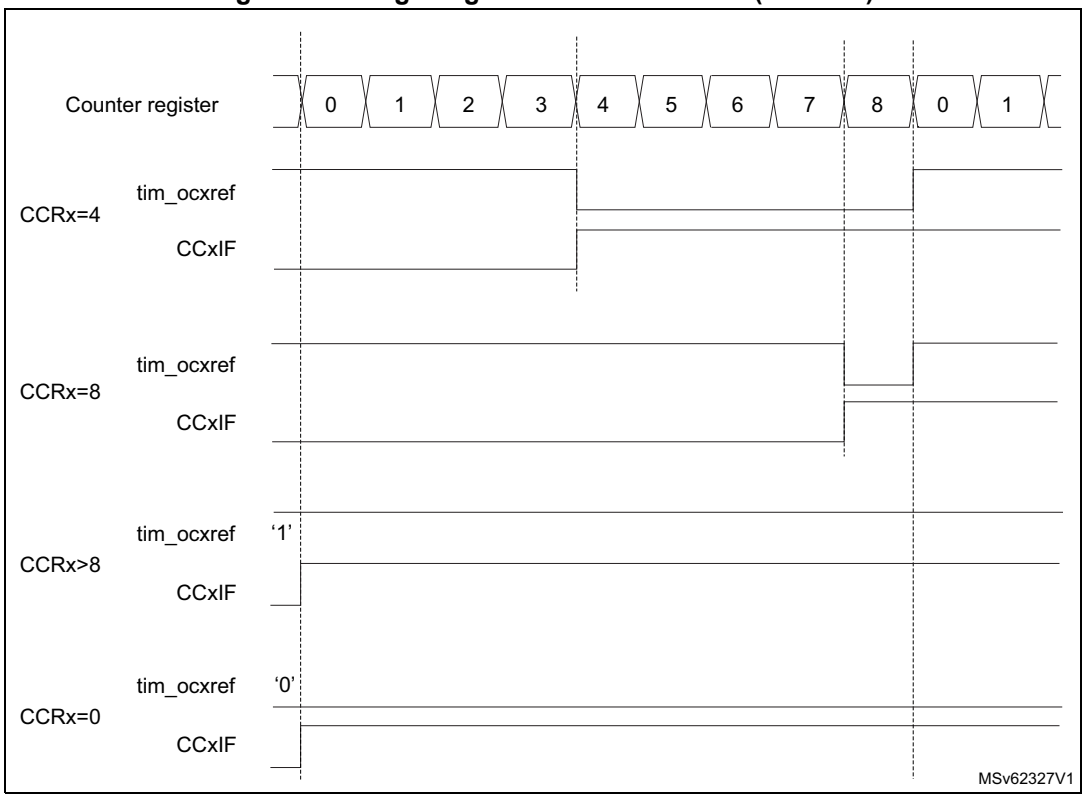
The tim_ocx polarity is software programmable using the CCxP bit in the TIM_CCER register. It can be programmed as active high or active low. The tim_ocx output is enabled by the CCxE bit in the TIM_CCER register. Refer to the TIM_CCERx register description for more details.

In PWM mode (1 or 2), TIM_CNT and TIM_CCRx are always compared to determine whether TIM_CNT ≤ TIM_CCRx.

The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

In the following example, the user considers PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIM_CNT < TIM_CCRx else it becomes low. If the compare value in TIM_CCRx is greater than the autoreload value (in TIM_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 747* shows some edge-aligned PWM waveforms in an example where TIM_ARR = 8.

Figure 747. Edge-aligned PWM waveforms (ARR = 8)

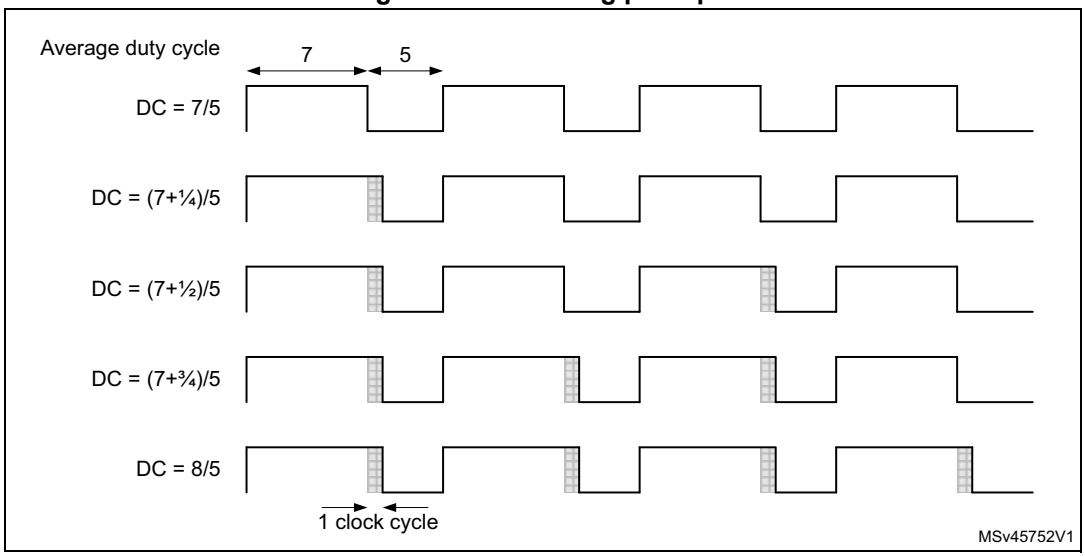


Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIM_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 748* presents the dithering principle applied to four consecutive PWM cycles.

Figure 748. Dithering principle

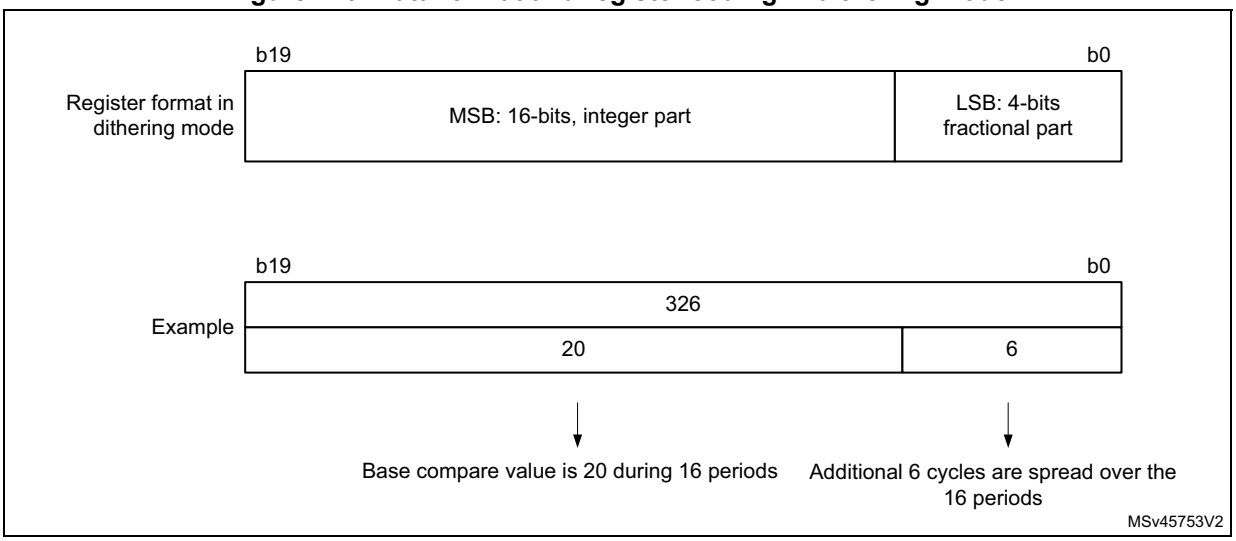


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 749* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The DITHEN bit must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 749. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

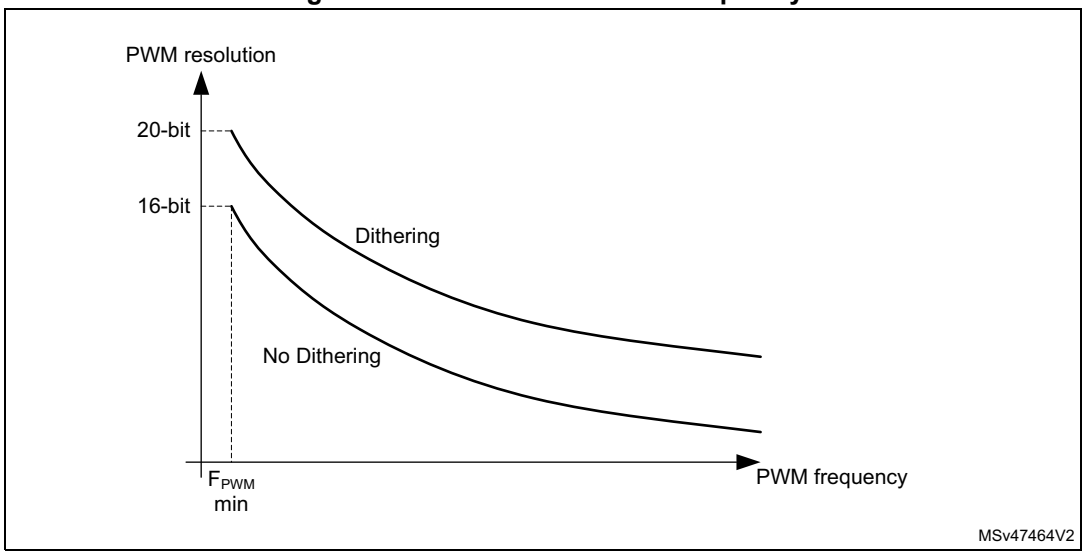
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIM_ARR and TIM_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

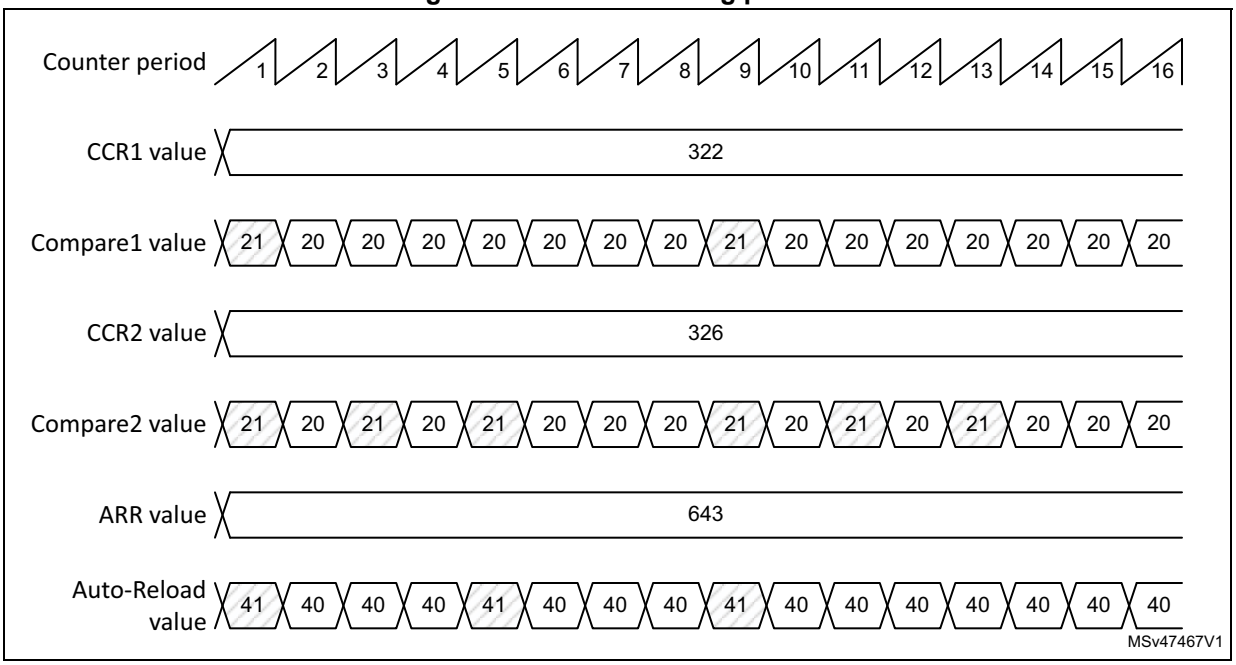
As shown on *Figure 750*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 750. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 751*.

Figure 751. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 424*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 424. CCR and ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-

Table 424. CCR and ARR register change dithering pattern (continued)

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

37.4.12 Combined PWM mode (TIM9/TIM12 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and delay are determined by the two TIM_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by the TIM_CCR1 and TIM_CCR2 registers.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIM_CCMRx register.

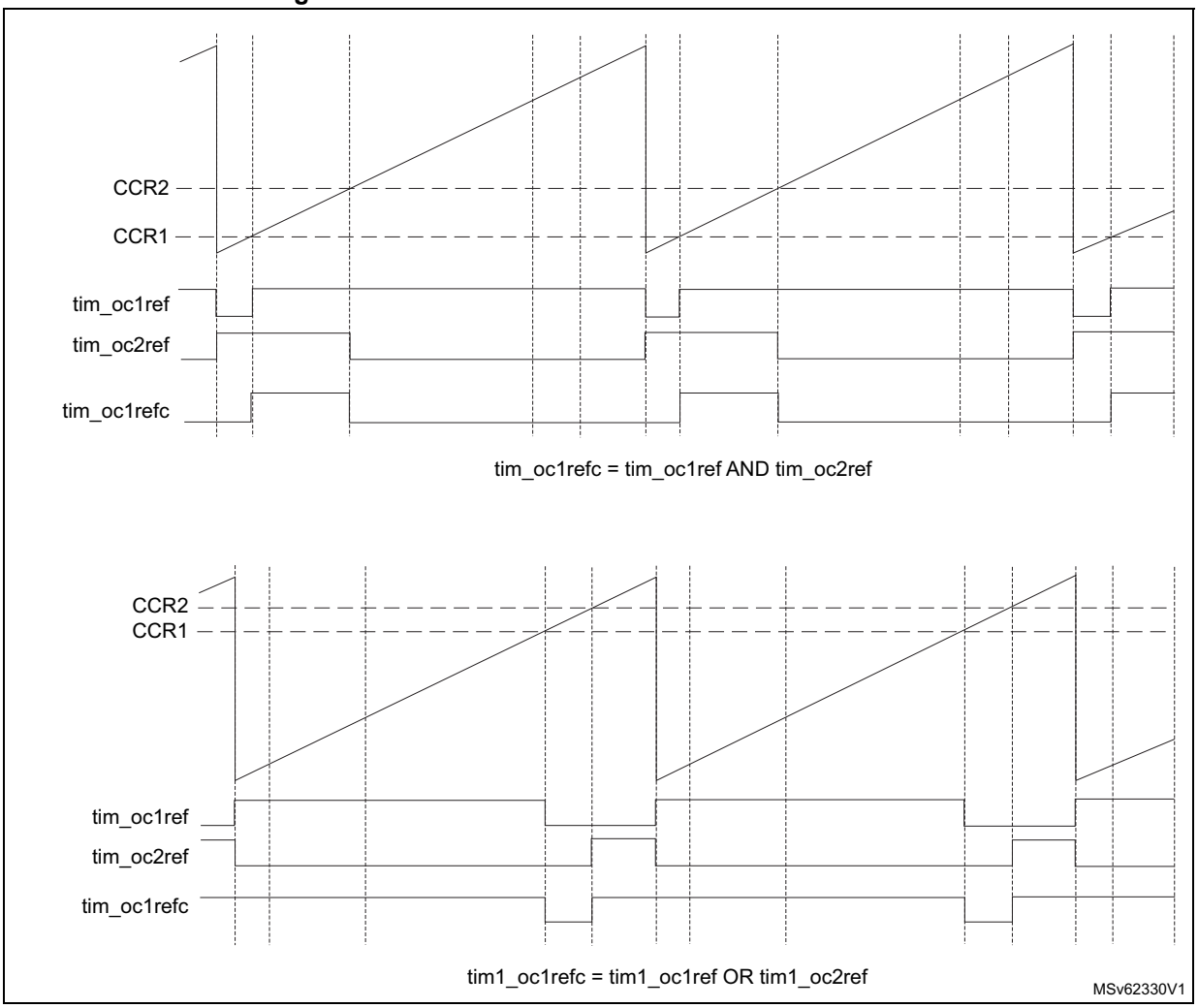
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 752 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.

Figure 752. Combined PWM mode on channel 1 and 2

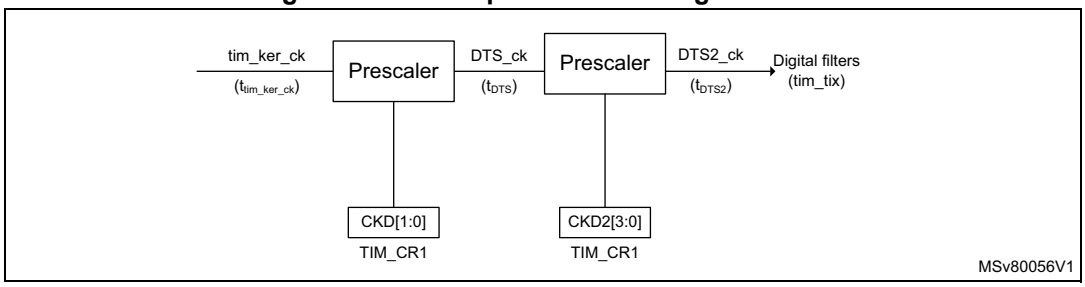


37.4.13 Clock dividers for digital filters

The digital filters (for tim_tix) are using a DTS_ck clock (with a t_{DTS} period), derived from the tim_ker_ck clock.

A prescaler can decrease the frequency to extend the DTS_ck clock period programmed with the CKD[1:0] bitfield in the TIM_CR1 register. A second prescaler can decrease the resulting DTS_ck clock to extend the digital filters range, with the CKD2[3:0] bitfield in the TIM_CR1 register.

Figure 753. Clock prescalers for digital filters



37.4.14 One-pulse mode

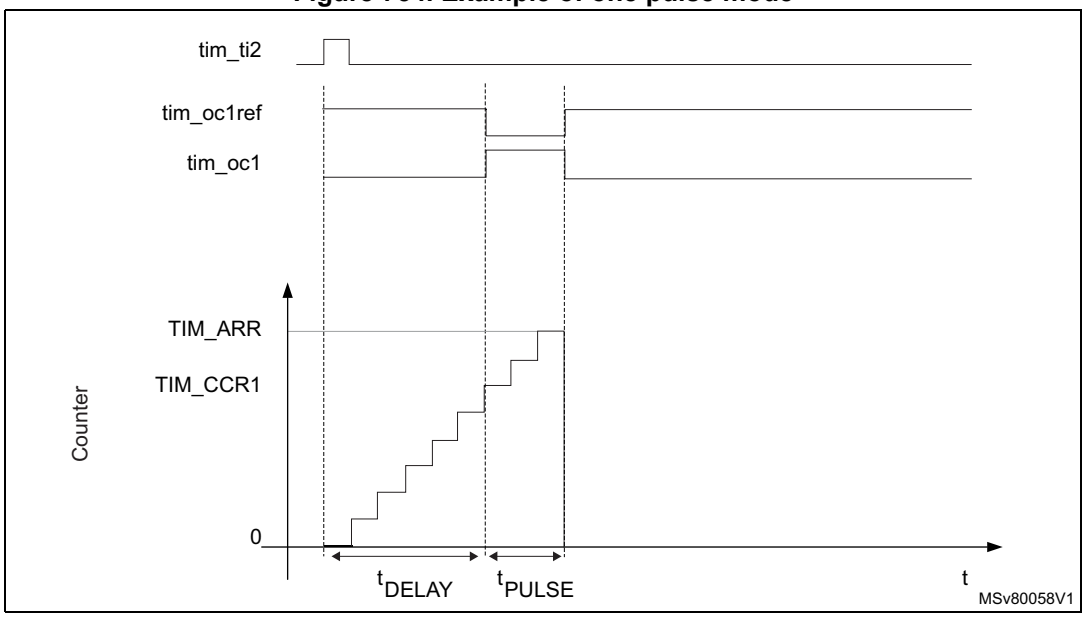
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIM_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be as follows:

CNT < CCRx ≤ ARR (in particular, 0 < CCRx)

Figure 754. Example of one pulse mode



For example one may want to generate a positive pulse on tim_oc1 with a length of tPULSE and after a delay of tDELAY as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIM_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIM_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIM_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIM_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIM_ARR - TIM_CCR1).
- Assuming the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 0111 in the TIM_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIM_CCMR1 register and ARPE in the TIM_CR1 register. In this case one has to write the compare value in the TIM_CCR1 register, the autoreload value in the TIM_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

Since only one pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIM_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0). When OPM bit in the TIM_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be obtained.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIM_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

37.4.15 Retriggerable one pulse mode (TIM9/TIM12 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with nonretriggerable one pulse mode described in *Section 37.4.14: One-pulse mode*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

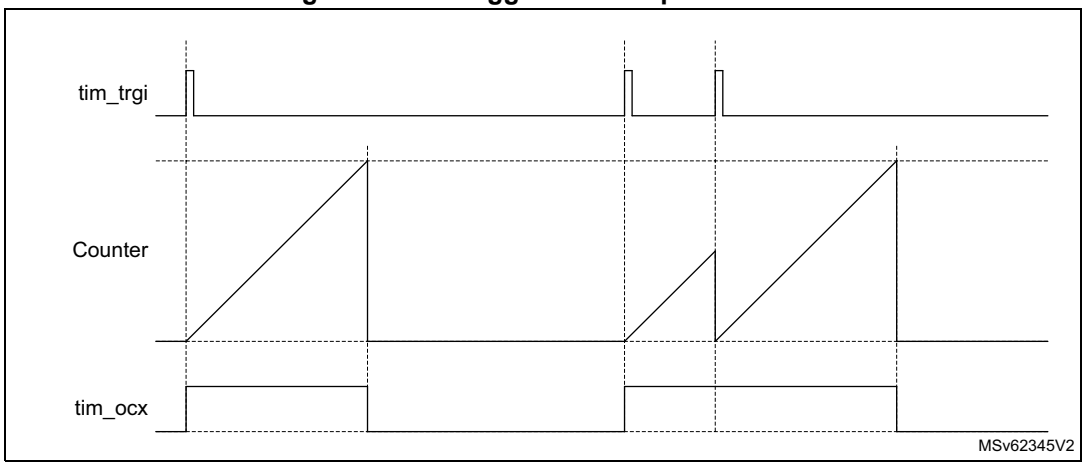
The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined reset + trigger mode) in the TIM_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for retriggerable OPM mode 1 or 2.

If the timer is configured in up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIM_CR1.

Figure 755. Retriggerable one pulse mode



37.4.16 UIF bit remapping

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag UIF into bit 31 of the timer counter register (TIM_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

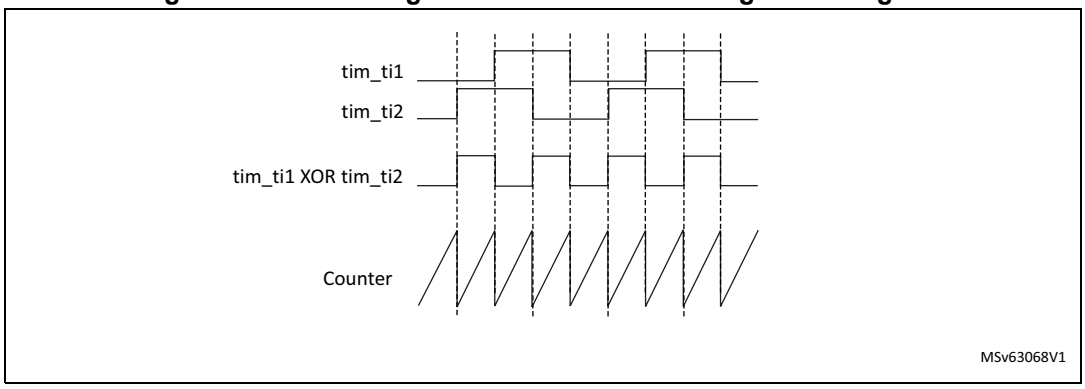
There is no latency between the assertions of the UIF and UIFCPY flags.

37.4.17 Timer input XOR function

The TI1S bit in the TIM_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 756*.

Figure 756. Measuring time interval between edges on 2 signals



The XORPS control bit selects the position of the XOR gate, as shown in *Figure 757* and *Figure 758*, when the TI1S bit is set.

Figure 757. XOR gate connection with TI1S = 1 and XORPS = 0

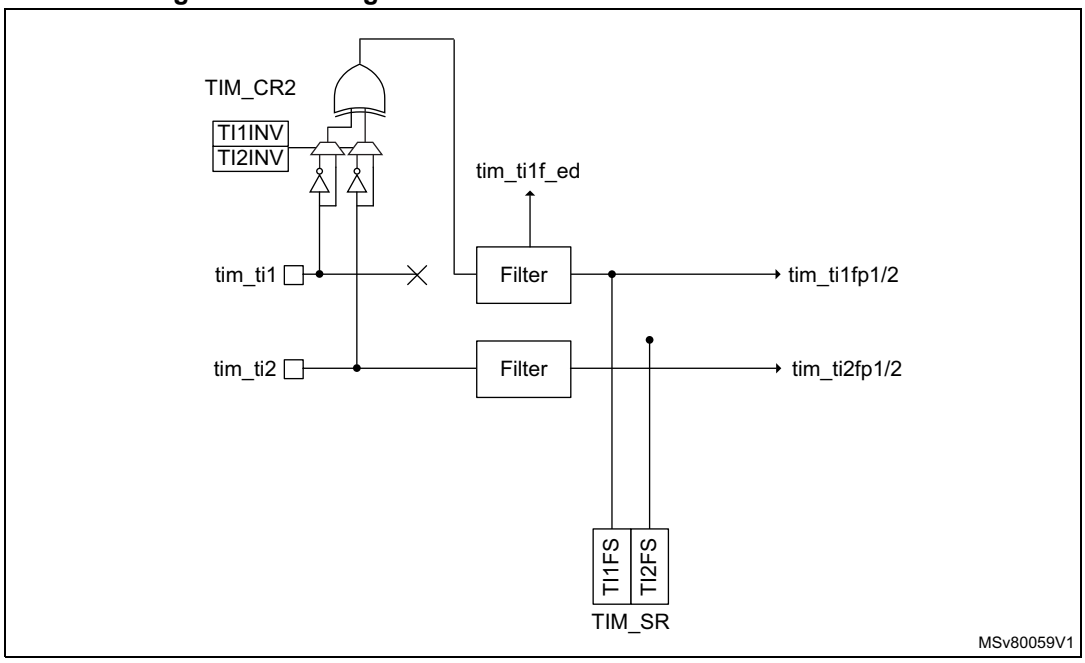
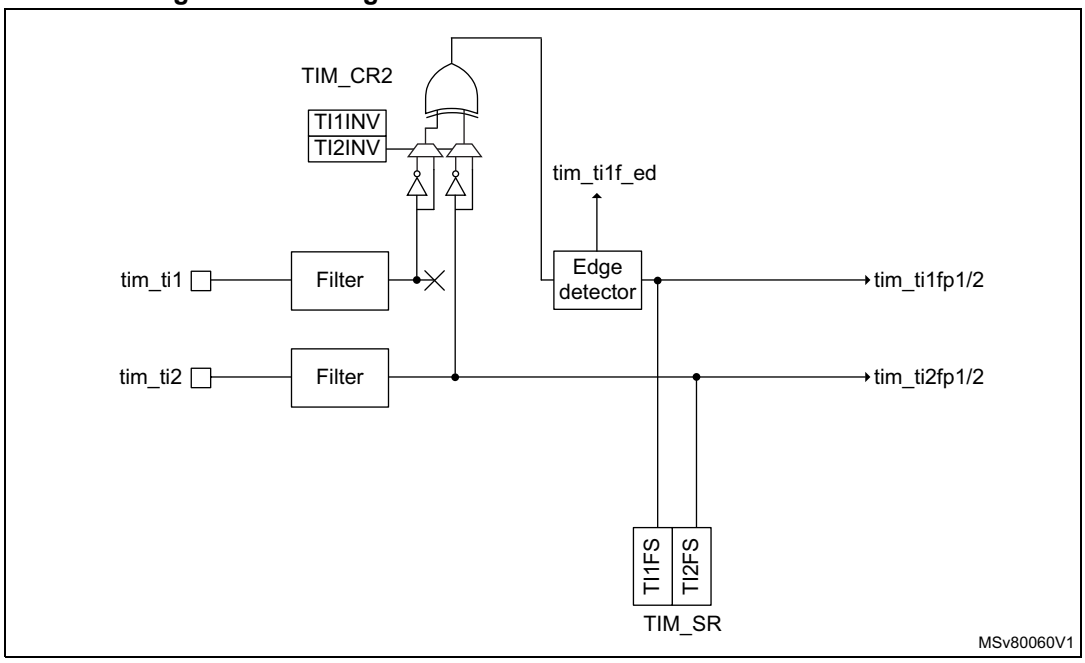


Figure 758. XOR gate connection with TI1S = 1 and XORPS = 1



Two control bits TI1INV and TI2INV in the TIM_CR2 register can invert the tim_in1 and tim_in2 signals prior entering the XOR gate.

The status of the tim_in1 and tim_in2 signals can be software polled using the TI1FS and TI2FS bits in the TIM_SR register.

37.4.18 TIM9/TIM12 external trigger synchronization

The TIM9/TIM12 timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and Gated + reset mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIM_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIM_ARR, TIM_CCRx) are updated.

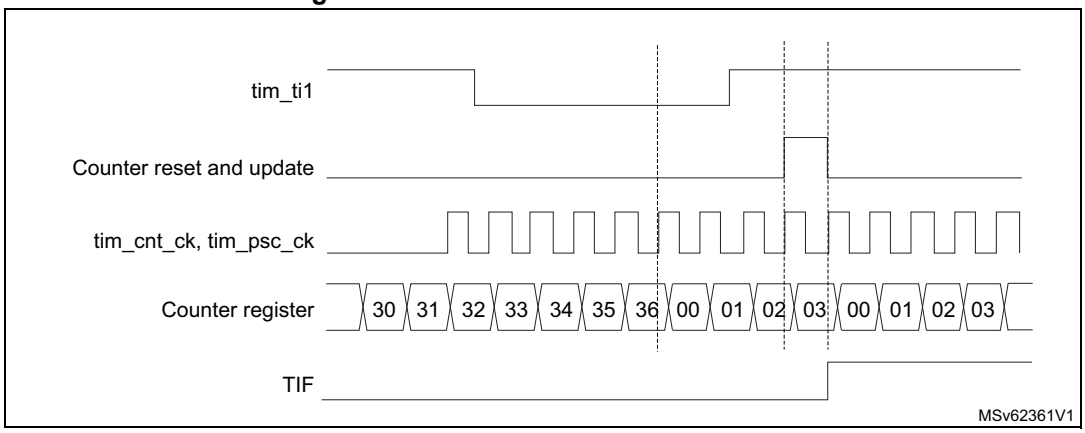
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIM_CCMR1 register. Program CC1P and CC1NP to 00 in TIM_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 100 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Start the counter by writing CEN = 1 in the TIM_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIM_SR register) and an interrupt request can be sent if enabled (depending on the TIE bit in TIM_DIER register).

The following figure shows this behavior when the autoreload register TIM_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 759. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

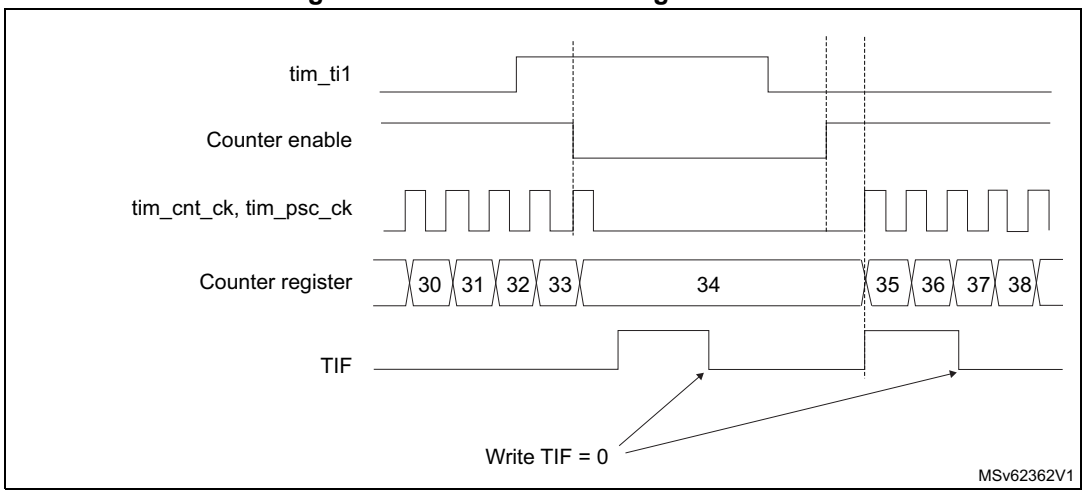
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIM_CCMR1 register. Program CC1P = 1 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 101 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIM_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIM_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 760. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

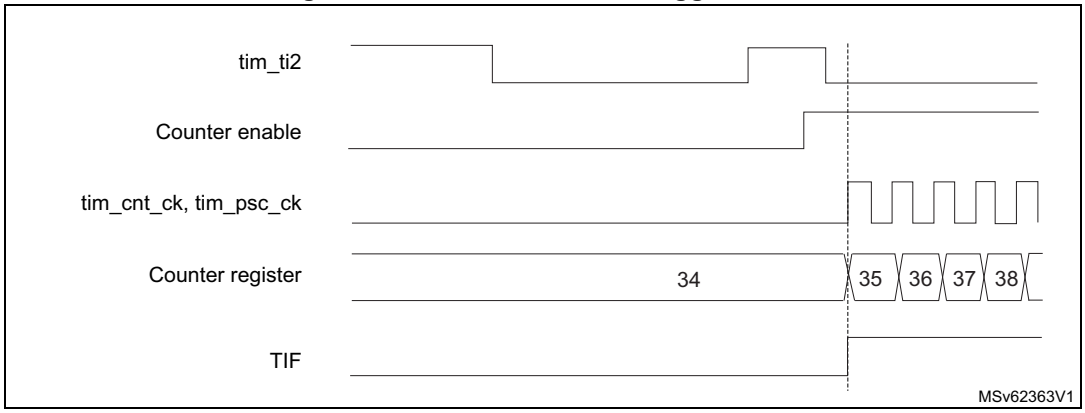
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIM_CCMR1 register. Program CC2P = 1 and CC2NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in TIM_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIM_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 761. Control circuit in trigger mode



37.4.19 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

37.4.20 Slave mode – combined reset + gated mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

37.4.21 Timer synchronization (TIM9/TIM12 only)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 35.4.27: Timer synchronization* for details.

Note: The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

37.4.22 Using timer output as trigger for other timers (TIM10/TIM11/TIM13/TIM14 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

37.4.23 ADC triggers (TIM9/TIM12 only)

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

37.4.24 ADC synchronization (TIM9/TIM12 only)

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIM_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- When the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.
- When the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source):

1. The counter period must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times (\text{TIM_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times \text{TIM_CCRx} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements is met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock
2. Configure the timer and set the ADSYNC bit
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits)
4. Start the timer (with the CEN counter enable bit)

37.4.25 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to the debug section.

37.5 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 low-power modes

Table 425. Effect of low-power modes on TIM9/TIM10/TIM11/TIM12/TIM13/TIM14

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

37.6 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 interrupts

The TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 can generate multiple interrupts, as shown in *Table 426*.

Table 426. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM	Update	UIF	UIE	write 0 in UIF	Yes	No
	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
TIM	Capture/compare 2(1)	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Trigger(1)	TIF	TIE	write 0 in TIF	Yes	No

1. Available for TIM9/TIM12 only.

37.7 TIM9/TIM12 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits), or words (32 bits).

37.7.1 TIM control register 1 (TIM_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CKD2[3:0]**: Clock division 2

This bit field indicates the division between sampling clock (t_{DTS}) and the digital filters (tim_tix).

0000: No Division
0001: $t_{DTS2} = 4 \times t_{DTS}$
0010: $t_{DTS2} = 16 \times t_{DTS}$
0011: $t_{DTS2} = 64 \times t_{DTS}$
0100: $t_{DTS2} = 256 \times t_{DTS}$
0101: $t_{DTS2} = 1024 \times t_{DTS}$
0110: $t_{DTS2} = 4096 \times t_{DTS}$
0111: $t_{DTS2} = 16384 \times t_{DTS}$
1000: $t_{DTS2} = 65536 \times t_{DTS}$
1001: $t_{DTS2} = 262144 \times t_{DTS}$
Others: Reserved, do not program this value.

Bits 23:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2 \times t_{tim_ker_ck}$
10: $t_{DTS} = 4 \times t_{tim_ker_ck}$
11: $t_{DTS} = 8 \times t_{tim_ker_ck}$

Bit 7 **ARPE**: Auto-reload preload enable

0: TIM_ARR register is not buffered.
1: TIM_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generates an update interrupt if enabled. These events can be:
– Counter overflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow generates an update interrupt if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable update event (UEV) generation.
0: UEV enabled. An UEV is generated by one of the following events:
– Counter overflow
– Setting the UG bit
Buffered registers are then loaded with their preload values.
1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
CEN is cleared automatically in one-pulse mode, when an update event occurs.
Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

37.7.2 TIM control register 2 (TIM_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TI2INV	TI1INV	AD SYNC	XORPS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			Res.	Res.	Res.	Res.
								rw	rw	rw	rw				

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TI2INV**: tim_ti2 signal inversion on XOR gate input
0: tim_ti2 non-inverted
1: tim_ti2 inverted

Bit 29 **TI1INV**: tim_ti1 signal inversion on XOR gate input
0: tim_ti1 non-inverted
1: tim_ti1 inverted

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 37.4.24* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bit 27 **XORPS**: XOR gate position selection

This bit defines the position of the XOR gate with respect to input filters.

0: XOR gate placed before TI1 filter

1: XOR gate placed after TI1 and TI2 filters

Bits 26:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: tim_ti1 selection

This bit enables the insertion of an XOR gate to combine tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs into tim_ti1.

The position of the XOR gate, with respect to input filters, is defined with the XORPS bit.

0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input

1: The tim_ti1_in[15:0] and tim_ti2_in[15:0] multiplexers output are connected to the tim_ti1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

000: Reset - the UG bit from the TIM_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

001: Enable - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM_SMCR register).

010: Update - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

011: Compare Pulse - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

100: Compare - tim_oc1refc signal is used as trigger output (tim_trgo).

101: Compare - tim_oc2refc signal is used as trigger output (tim_trgo).

Bits 3:0 Reserved, must be kept at reset value.

37.7.3 TIM slave mode control register (TIM_SMCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw

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

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

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/Slave mode

0: No action

1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful in order to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This TS[4:0] bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal trigger 0 (tim_itr0)
00001: Internal trigger 1 (tim_itr1)
00010: Internal trigger 2 (tim_itr2)
00011: Internal trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: Reserved
01000: Internal trigger 4 (tim_itr4)
01001: Internal trigger 5 (tim_itr5)
01010: Internal trigger 6 (tim_itr6)
01011: Internal trigger 7 (tim_itr7)
01100: Internal trigger 8 (tim_itr8)
01101: Internal trigger 9 (tim_itr9)
01110: Internal trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
10100: Internal trigger 16 (tim_itr16)
10101: Internal trigger 17 (tim_itr17)
10110: Internal trigger 18 (tim_itr18)
10111: Internal trigger 19 (tim_itr19)
11000: Internal trigger 20 (tim_itr20)
11001: Internal trigger 21 (tim_itr21)
11010: Internal trigger 22 (tim_itr22)
11011: Internal trigger 23 (tim_itr23)
11100: Internal trigger 24 (tim_itr24)
11101: Internal trigger 25 (tim_itr25)
11110: Internal trigger 26 (tim_itr26)
11111: Internal trigger 27 (tim_itr27)

See *Section 37.4.2: TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 pins and internal signals* for more details on the meaning of tim_itrx for each timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (see input control register and control register description.

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Reserved

0010: Reserved

0011: Reserved

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

Other codes: reserved.

Note: The gated mode (including gated + reset mode) must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signals must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

37.7.4 TIM interrupt enable register (TIM_DIER)

Address offset: 0x00C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
									rw				rw	rw	rw

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

Bit 6 **TIE**: Trigger interrupt enable

0: Trigger interrupt disabled.

1: Trigger interrupt enabled.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable

0: CC2 interrupt disabled.

1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

37.7.5 TIM status register (TIM_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		r	r				rc_w0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **TI2FS**: tim_ti2f status
This flag indicates the value of the tim_ti2f signal (after the digital filtering stage), for polling purpose.
0: tim_ti2f signal is low
1: tim_ti2f signal is high

Bit 28 **TI1FS**: tim_ti1f status
This flag indicates the value of the tim_ti1f signal (after the digital filtering stage), for polling purpose.
0: tim_ti1f signal is low
1: tim_ti1f signal is high

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **UIOVRF**: Update interrupt overrun flag
This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.
0: No update interrupt overrun
1: Update interrupt overrun

Bits 23:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIM_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIM_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred.
If channel CC1 is configured as output: this flag is set when he content of the counter TIM_CNT matches the content of the TIM_CCR1 register. When the content of TIM_CCR1 is greater than the content of TIM_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIM_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIM_CCR1 register (an edge has been detected on tim_ic1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIM_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow and if UDIS = 0 in the TIM_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.
– When CNT is reinitialized by a trigger event (refer to *Section 37.7.3: TIM slave mode control register (TIM_SMCR)*), if URS = 0 and UDIS = 0 in the TIM_CR1 register.

37.7.6 TIM event generation register (TIM_EGR)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
									w				w	w	w

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

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in the TIM_SR register. Related interrupt can occur if enabled

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
the CC1IF flag is set, the corresponding interrupt is sent if enabled.
If channel CC1 is configured as input:
The current counter value is captured in the TIM_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initializes the counter and generates an update of the registers. The prescaler counter is also cleared and the prescaler ratio is not affected. The counter is cleared.

37.7.7 TIM capture/compare mode register 1 (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on tim_ti2
10: CC2 channel is configured as input, IC2 is mapped on tim_ti1
11: CC2 channel is configured as input, IC2 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIM_SMCR register)
Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample the tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIM_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

37.7.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

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

Bit 15 Reserved, must be kept at reset value.

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
Refer to OC1M[3:0] for bit description.

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on tim_ti2
10: CC2 channel is configured as input, IC2 is mapped on tim_ti1
11: CC2 channel is configured as input, IC2 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIM_SMCR register
Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bit 7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas the active level of tim_oc1 depends on the CC1P.

0000: Frozen - The comparison between the output compare register TIM_CCR1 and the counter TIM_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. The tim_oc1ref signal is forced high when the TIM_CNT counter matches the capture/compare register 1 (TIM_CCR1).

0010: Set channel 1 to inactive level on match. The tim_oc1ref signal is forced low when the TIM_CNT counter matches the capture/compare register 1 (TIM_CCR1).

0011: Toggle - tim_oc1ref toggles when TIM_CNT = TIM_CCR1

0100: Force inactive level - tim_oc1ref is forced low

0101: Force active level - tim_oc1ref is forced high

0110: PWM mode 1 - channel 1 is active as long as TIM_CNT<TIM_CCR1 else it is inactive

0111: PWM mode 2 - channel 1 is inactive as long as TIM_CNT<TIM_CCR1 else it is active

1000: Retrigerrable OPM mode 1 - The channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1001: Retrigerrable OPM mode 2 - The channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Reserved,

1111: Reserved

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIM_CCR1 disabled. TIM_CCR1 can be written at anytime, the new value is taken into account immediately

1: Preload register on TIM_CCR1 enabled. Read/Write operations access the preload register. TIM_CCR1 preload value is loaded into the active register at each update event

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on the counter and CCR1 values even when the trigger is ON. The minimum delay to activate the CC1 output when an edge occurs on the trigger input is 5 clock cycles

1: An active edge on the trigger input acts like a compare match on the CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00:CC1 channel is configured as output
01:CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10:CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11:CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIM_SMCR register)
Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

37.7.9 TIM capture/compare enable register (TIM_CCER)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 output Polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output Polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity
CC1 channel configured as output: CC1NP must be kept cleared
CC1 channel configured as input: CC1NP is used in conjunction with CC1P to define tim_ti1fp1/tim_ti2fp1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.

0: tim_oc1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: tim_oc1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of tim_ti1fp1 and tim_ti2fp1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to tim_tixfp1 rising edge (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger operation in gated mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to tim_tixfp1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both tim_tixfp1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/Compare 1 output enable.

0: Capture mode disabled / tim_oc1 is not active

1: Capture mode enabled / tim_oc1 signal is output on the corresponding output pin

Table 427. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The states of the external I/O pins connected to the standard tim_ocx *channels depend on the state of the* tim_ocx *channel and on the GPIO registers.*

37.7.10 TIM counter (TIM_CNT)

Address offset: 0x024

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIM_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

37.7.11 TIM prescaler (TIM_PSC)

Address offset: 0x028

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIM_EGR register or through trigger controller when configured in “reset mode”).

37.7.12 TIM autoreload register (TIM_ARR)

Address offset: 0x02C

Reset value: 0x0000 FFFF

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 37.4.3: Time-base unit* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.
Non-dithering mode (DITHEN = 0)
The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

37.7.13 TIM capture/compare register 1 (TIM_CCR1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIM_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

37.7.14 TIM capture/compare register 2 (TIM_CCR2)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output:
CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIM_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIM_CNT and signalled on tim_oc2 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.
If channel CC2 is configured as input:
CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIM_CCR2 register is read-only and cannot be programmed.
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

37.7.15 TIM timer input selection register (TIM_TISEL)

Address offset: 0x05C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

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

Bits 11:8 **TI2SEL[3:0]**: selects tim_ti2_in[15:0] input
0000: TIM_CH2 input (tim_ti2_in0)
0001: tim_ti2_in1
...
0100: tim_ti2_in15
Refer to *Section 37.4.2: TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIM_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 37.4.2: TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 pins and internal signals* for interconnects list.

37.7.16 TIM9/TIM12 register map

TIM9/TIM12 registers are mapped as 16-bit addressable registers as described below:

Table 428. TIM9/TIM12 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**TIM_CR1**	Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value					0	0	0	0												0	0		0	0	0				0	0	0	0
0x004	**TIM_CR2**	Res.	TI2INV	TI1INV	ADSYNC	XORPS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			Res.	Res.	Res.	Res.
	Reset value		0	0	0	0																				0	0	0	0				
0x008	**TIM_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0									0	0	0	0		0	0	0
0x00C	**TIM_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value								0														0	0			0				0	0	0
0x010	**TIM_SR**	Res.	Res.	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value			0	0				0														0	0			0				0	0	0
0x014	**TIM_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
	Reset value																										0				0	0	0
0x018	**TIM_CCMR1** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Res.	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value								0								0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x01C	Reserved	Res.																															
0x020	**TIM_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																									0		0	0	0		0	0
0x024	**TIM_CNT**	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**TIM_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 428. TIM9/TIM12 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	**TIM_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	Reserved	Res.																															
0x034	**TIM_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	**TIM_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C to 0x058	Reserved	Res.																															
0x05C	**TIM_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0
0x060 - 0x3E8	Reserved	Res.																															

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

37.8 TIM10/TIM11/TIM13/TIM14 registers

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits), or words (32 bits).

37.8.1 TIM control register 1 (TIM_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CKD2[3:0]**: Clock division 2

This bit field indicates the division between sampling clock (t_{DTS}) and the digital filters (tim_tix).
0000: No Division
0001: $t_{DTS2} = 4 \times t_{DTS}$
0010: $t_{DTS2} = 16 \times t_{DTS}$
0011: $t_{DTS2} = 64 \times t_{DTS}$
0100: $t_{DTS2} = 256 \times t_{DTS}$
0101: $t_{DTS2} = 1024 \times t_{DTS}$
0110: $t_{DTS2} = 4096 \times t_{DTS}$
0111: $t_{DTS2} = 16384 \times t_{DTS}$
1000: $t_{DTS2} = 65536 \times t_{DTS}$
1001: $t_{DTS2} = 262144 \times t_{DTS}$
Others: Reserved, do not program this value.

Bits 23:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_tix),
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2 \times t_{tim_ker_ck}$
10: $t_{DTS} = 4 \times t_{tim_ker_ck}$
11: $t_{DTS} = 8 \times t_{tim_ker_ck}$

Bit 7 **ARPE**: Auto-reload preload enable
0: TIM_ARR register is not buffered
1: TIM_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the update interrupt (UEV) sources.
0: Any of the following events generate an UEV if enabled:
– Counter overflow
– Setting the UG bit
1: Only counter overflow generates an UEV if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable update interrupt (UEV) event generation.
0: UEV enabled. An UEV is generated by one of the following events:
– Counter overflow
– Setting the UG bit.
Buffered registers are then loaded with their preload values.
1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

37.8.2 TIM interrupt enable register (TIM_DIER)

Address offset: 0x00C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IE	UIE
														rw	rw

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

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

37.8.3 TIM status register (TIM_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TI1FS	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			r				rc_w0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IF	UIF
						rc_w0								rc_w0	rc_w0

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **TI1FS**: tim_ti1f status

This flag indicates the value of the tim_ti1f signal (after the digital filtering stage), for polling purpose.

0: tim_ti1f signal is low

1: tim_ti1f signal is high

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **UIOVRF**: Update Interrupt Overrun flag

This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.

0: No Update Interrupt Overrun

1: Update Interrupt Overrun

Bits 23:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected.

1: The counter value has been captured in TIM_CCR1 register while CC1IF flag was already set

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIM_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred.

If channel CC1 is configured as output: this flag is set when he content of the counter TIM_CNT matches the content of the TIM_CCR1 register. When the content of TIM_CCR1 is greater than the content of TIM_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIM_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIM_CCR1 register (an edge has been detected on tim_ic1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIM_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow and if UDIS = 0 in the TIM_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.

37.8.4 TIM event generation register (TIM_EGR)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1G	UG
														w	w

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

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIM_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared.

37.8.5 TIM capture/compare mode register 1 (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:
0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIM_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: Reserved
11: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

37.8.6 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
									rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

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

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.
0000: Frozen. The comparison between the output compare register TIM_CCR1 and the counter TIM_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.
0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).
0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).
0011: Toggle - tim_oc1ref toggles when TIM_CNT = TIM_CCR1.
0100: Force inactive level - tim_oc1ref is forced low.
0101: Force active level - tim_oc1ref is forced high.
0110: PWM mode 1 - Channel 1 is active as long as TIM_CNT < TIM_CCR1 else inactive.
0111: PWM mode 2 - Channel 1 is inactive as long as TIM_CNT < TIM_CCR1 else active
Others: Reserved

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIM_CCR1 disabled. TIM_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR1 enabled. Read/Write operations access the preload register. TIM_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. OC is then set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.

10: Reserved.

11: Reserved.

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

37.8.7 TIM capture/compare enable register (TIM_CCER)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	Res.	CC1P	CC1E
												rw		rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity.

CC1 channel configured as output: CC1NP must be kept cleared.

CC1 channel configured as input: CC1NP bit is used in conjunction with CC1P to define tim_ti1fp1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.
0: tim_oc1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: tim_oc1 active low (output mode) / Edge sensitivity selection (input mode, see below)
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of tim_ti1fp1 for capture operations.
CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to tim_ti1fp1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).
CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to tim_ti1fp1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_ti1fp1 is inverted (trigger operation in gated mode).
CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both tim_ti1fp1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_ti1fp1 not inverted (trigger operation in gated mode).
CC1NP = 1, CC1P = 0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/Compare 1 output enable.
0: Capture mode disabled / tim_oc1 is not active
1: Capture mode enabled / tim_oc1 signal is output on the corresponding output pin

Table 429. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output Disabled (tim_ocx = 0)
1	tim_ocx = tim_ocxref + Polarity

Note: The state of the external I/O pins connected to the standard tim_ocx channels depends on the tim_ocx channel state and the GPIO registers.

37.8.8 TIM counter (TIM_CNT)

Address offset: 0x024

Reset value: 0x0000 0000

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

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

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit in the TIM_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

37.8.9 TIM prescaler (TIM_PSC)

Address offset: 0x028

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event.

(including when the counter is cleared through UG bit of TIM_EGR register or through trigger controller when configured in “reset mode”).

37.8.10 TIM autoreload register (TIM_ARR)

Address offset: 0x02C

Reset value: 0x0000 FFFF

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 37.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

37.8.11 TIM capture/compare register 1 (TIM_CCR1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIM_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

37.8.12 TIM timer input selection register (TIM_TISEL)

Address offset: 0x05C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIM_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 37.4.2: TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 pins and internal signals* for interconnects list.

37.8.13 TIM10/TIM11/TIM13/TIM14 register map

TIM10/TIM11/TIM13/TIM14 registers are mapped as 16-bit addressable registers as described in the tables below:

Table 430. TIM10/TIM11/TIM13/TIM14 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**TIM_CR1**	Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value					0	0	0	0												0	0		0	0	0				0	0	0	0
0x004 to 0x008	Reserved	Res.																															
0x00C	**TIM_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IE	UIE
	Reset value																															0	0
0x010	**TIM_SR**	Res.	Res.	Res.	TI1FS	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IF	UIF
	Reset value				0				0															0								0	0
0x014	**TIM_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1G	UG
	Reset value																															0	0
0x018	**TIM_CCMR1** Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																0										0	0	0	0	0	0	0
	TIM_CCMR1 Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																										0	0	0	0	0	0	0	0
0x01C	Reserved	Res.																															
0x020	**TIM_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	Res.	CC1P	CC1E
	Reset value																													0		0	0

Table 430. TIM10/TIM11/TIM13/TIM14 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024	**TIM_CNT**	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**TIM_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	**TIM_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	Reserved	Res.																															
0x034	**TIM_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038 to 0x058	Reserved	Res.																															
0x05C	**TIM_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x060 - 0x3E8	Reserved	Res.																															

Refer to *Section 2.3* for the register boundary addresses.

38 General purpose timers (TIM15/TIM16/TIM17)

38.1 TIM15/TIM16/TIM17 introduction

The TIM15/TIM16/TIM17 timers consist of a 16-bit autoreload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16/TIM17 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in *Section 38.4.33: Timer synchronization (TIM15 only)*.

38.2 TIM15 main features

TIM15 includes the following features:

- 16-bit up, down, up/down autoreload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Two independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)
- Incremental encoders and quadrature encoders and hall sensors support
- ADC synchronization for jitter-free sampling points

38.3 TIM16/TIM17 main features

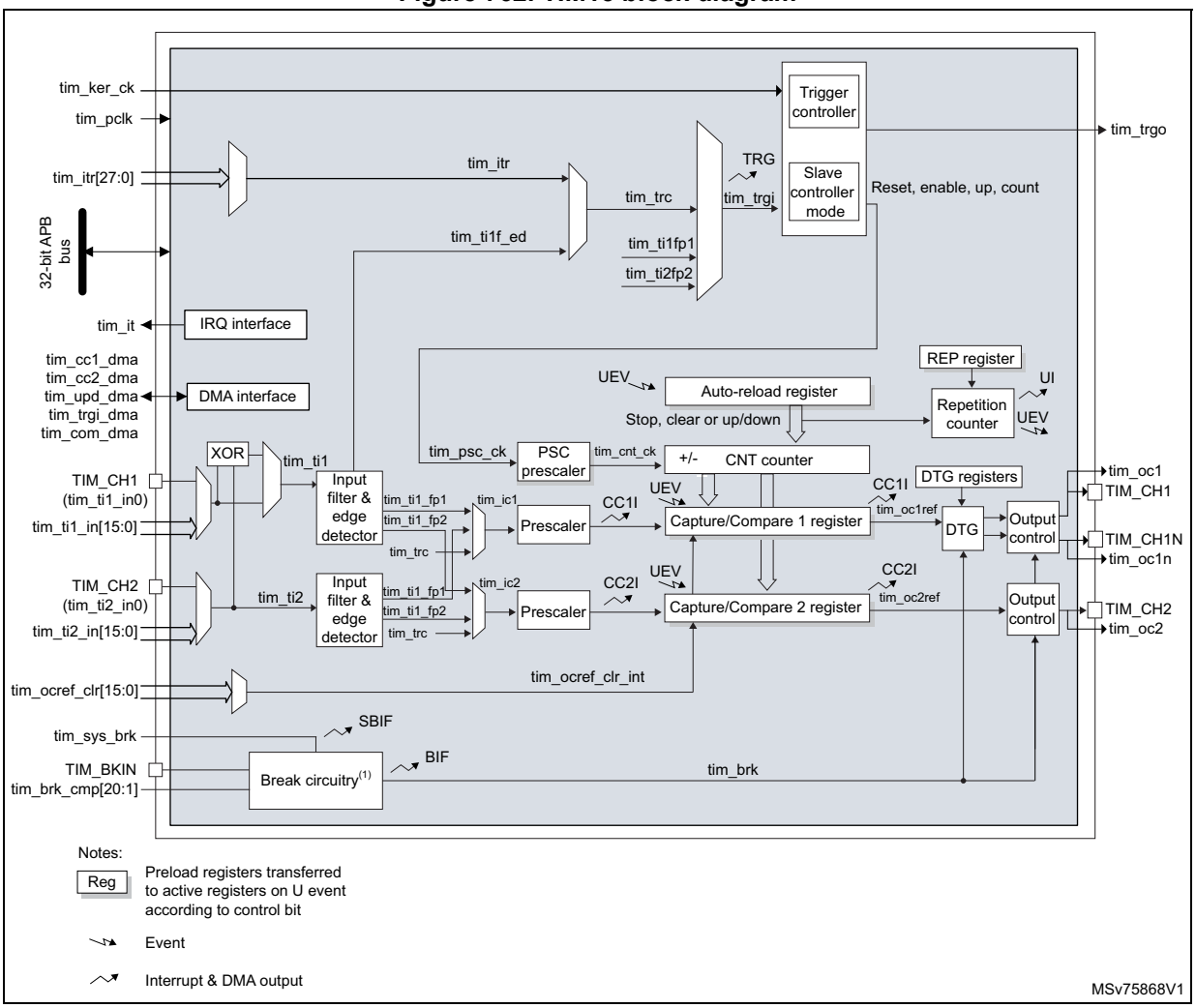
The TIM16/TIM17 timers include the following features:

- 16-bit up, down, up/down autoreload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input

38.4 TIM15/TIM16/TIM17 functional description

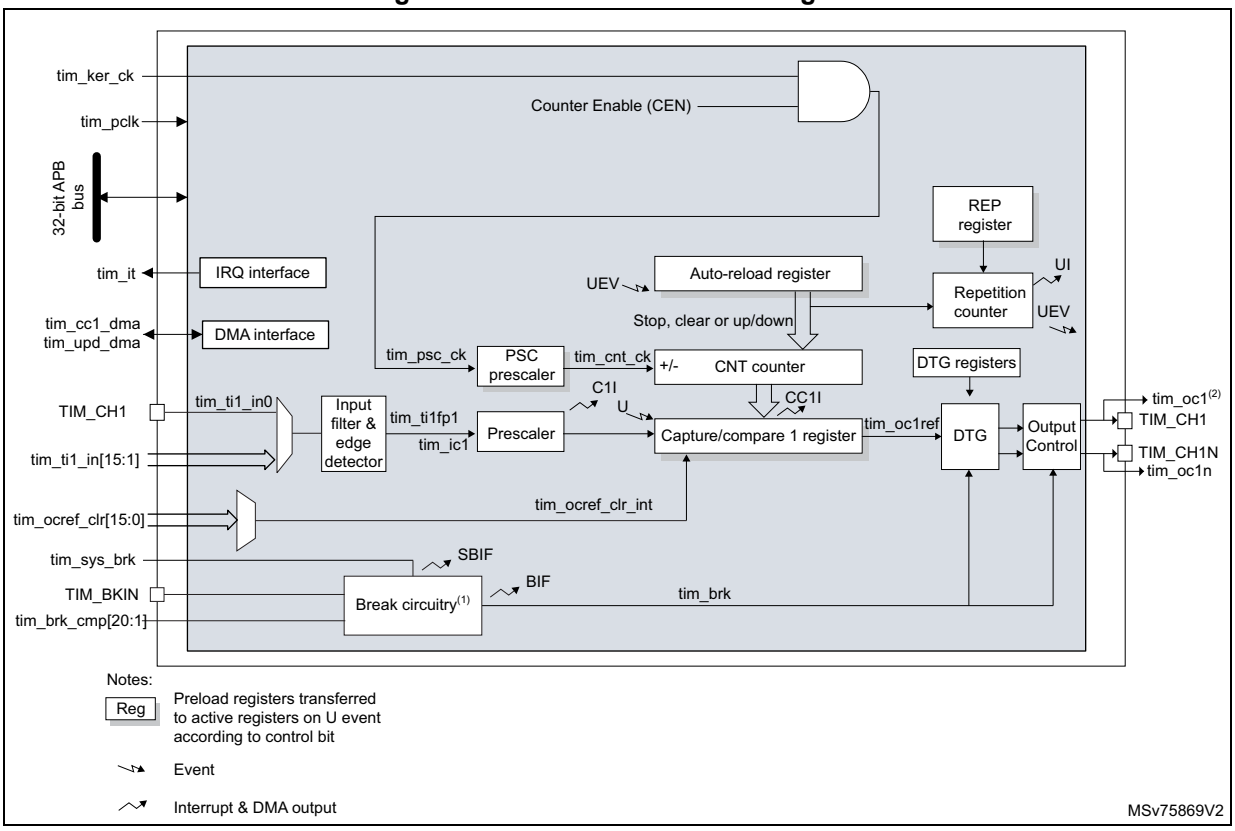
38.4.1 Block diagram

Figure 762. TIM15 block diagram



1. Refer to *Section 38.4.17: Using the break function* for details.

Figure 763. TIM16/TIM17 block diagram



1. Refer to *Section 38.4.17: Using the break function* for details.
2. This signal can be used as trigger for some slave timer (see internal trigger connection table in next section). See *Section 38.4.34: Using timer output as trigger for other timers (TIM16/TIM17 only)* for details.

38.4.2 TIM15/TIM16/TIM17 pins and internal signals

Table 431 and *Table 432* in this section summarize the TIM inputs and outputs.

Table 431. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs.
TIM_CH1N	Output	Timer complementary outputs, derived from TIM_CH1 output with the possibility to have deadtime insertion.
TIM_BKIN	Input / Output	Break input. This input can also be configured in bidirectional mode.

Table 432. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0]	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_itr[27:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocx (x = 1 to 2)	Output	Internal output compare signals of channels 1 to 2. These outputs can trigger or synchronize other internal peripherals.
tim_oc1n	Output	Internal timer complementary output derived from tim_oc1 internal output, with the option to insert dead time.
tim_ocref_clr[15:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[20:1]	Input	Break input for internal signals
tim_sys_brk[5:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests
tim_cc1_dma	Output	Timer capture / compare 1 dma request
tim_cc2_dma	Output	Timer capture / compare 2 dma request
tim_upd_dma	Output	Timer update dma request
tim_trgi_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

Table 433 and *Table 434* list the sources connected to the tim_ti[2:1] input multiplexers.

Table 433. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources		
	TIM15	TIM16	TIM17
tim_ti1_in0	TIM15_CH1	TIM16_CH1	TIM17_CH1
tim_ti1_in1	comp1_out	comp6_out	comp5_out
tim_ti1_in2	RCC_MCO1	RCC_LSI	RCC_HSE
tim_ti1_in3	RCC_MCO2	RCC_LSE	RCC_MCO1
tim_ti1_in4	RCC_HSI/1024	RTC_WKUP_IT	-
tim_ti1_in5	-	-	-
tim_ti1_in6	comp4_out	-	-
tim_ti1_in7	comp5_out	-	-
tim_ti1_in8	comp7_out	-	-
tim_ti1_in9	-	-	-
tim_ti1_in10	RCC_LSE	-	-
tim_ti1_in[15:11]	Reserved		

Table 434. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources
	TIM15
tim_ti2_in0	TIM15_CH2
tim_ti2_in1	comp2_out
tim_ti2_in2	comp3_out
tim_ti2_in3	comp6_out
tim_ti2_in4	comp8_out
tim_ti2_in[8:5]	Reserved
tim_ti2_in9	-
tim_ti2_in[15:10]	Reserved

Table 435 lists the internal sources connected to the tim_itr input multiplexer.

Table 435. TIMx internal trigger connection

tim_itrx inputs	TIM15
tim_itr0	tim1_trgo
tim_itr1	tim1_trgo2
tim_itr2	tim2_trgo

Table 435. TIMx internal trigger connection (continued)

tim_itrx inputs	TIM15
tim_itr3	tim3_trgo
tim_itr4	tim4_trgo
tim_itr5	tim5_trgo
tim_itr6	tim8_trgo
tim_itr7	tim8_trgo2
tim_itr8	tim9_trgo
tim_itr9	tim10_oc1
tim_itr10	tim11_oc1
tim_itr11	tim12_trgo
tim_itr12	tim13_oc1
tim_itr13	tim14_oc1
tim_itr14	-
tim_itr15	tim16_oc1
tim_itr16	tim17_oc1
tim_itr17	tim20_trgo
tim_itr18	tim20_trgo2
tim_itr19	-
tim_itr20	-
tim_itr21	hrtim1_sync_out2
tim_itr22	hrtim2_sync_out2
tim_itr[27:23]	Reserved

Table 436 and *Table 437* list the sources connected to the tim_brk input.

Table 436. Timer break interconnect

tim_brk inputs	TIM15	TIM16	TIM17
tim_brk_cmp0	TIM15_BKIN pin	TIM16_BKIN pin	TIM17_BKIN pin
tim_brk_cmp1	comp1_out	comp1_out	comp1_out
tim_brk_cmp2	comp2_out	comp2_out	comp2_out
tim_brk_cmp3	comp3_out	comp3_out	comp3_out
tim_brk_cmp4	comp4_out	comp4_out	comp4_out
tim_brk_cmp5	comp5_out	comp5_out	comp5_out
tim_brk_cmp6	comp6_out	comp6_out	comp6_out
tim_brk_cmp7	comp7_out	comp7_out	comp7_out

Table 436. Timer break interconnect (continued)

tim_brk inputs	TIM15	TIM16	TIM17
tim_brk_cmp8	comp8_out	comp8_out	comp8_out
tim_brk_cmp9	TIM1_BKIN2	TIM1_BKIN2	TIM1_BKIN2
tim_brk_cmp10	TIM8_BKIN2	TIM8_BKIN2	TIM8_BKIN2
tim_brk_cmp11	TIM20_BKIN2	TIM20_BKIN2	TIM20_BKIN2
tim_brk_cmp12	-	-	-
tim_brk_cmp13	hsp1_brk_out2	hsp1_brk_out2	hsp1_brk_out2
tim_brk_cmp14	hsp1_brk_out3	hsp1_brk_out3	hsp1_brk_out3
tim_brk_cmp15	TIM1_BKIN1	TIM1_BKIN1	TIM1_BKIN1
tim_brk_cmp16	TIM8_BKIN1	TIM8_BKIN1	TIM8_BKIN1
tim_brk_cmp17	TIM20_BKIN1	TIM20_BKIN1	TIM20_BKIN1
tim_brk_cmp18	-	TIM15_BKIN1	TIM15_BKIN1
tim_brk_cmp19	TIM16_BKIN1	-	TIM16_BKIN1
tim_brk_cmp20	TIM17_BKIN1	TIM17_BKIN1	-

Table 437. System break interconnect

tim_sys_brk inputs	TIM15 / TIM16 / TIM17	Enable bit in SBS_CFGR2 register
tim_sys_brk0	Cortex®-M7 LOCKUP	CLL
tim_sys_brk1	Programmable Voltage Detector (PVD)	PVDL
tim_sys_brk2	SRAM double ECC error	SEL
tim_sys_brk3	Flash double ECC error	ECCL
tim_sys_brk4	Clock Security System (CSS)	None (always enabled)
tim_sys_brk5	Cortex-M33 data cache double ECC error	DCECCL

Table 438 lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 438. Interconnect to the ocref_clr input multiplexer

Timer OCREF clear signal	Timer OCREF clear signals assignment		
	TIM15	TIM16	TIM17
tim_ocref_clr0	comp1_out	comp1_out	comp1_out
tim_ocref_clr1	comp2_out	comp2_out	comp2_out
tim_ocref_clr2	comp3_out	comp3_out	comp3_out
tim_ocref_clr3	comp4_out	comp4_out	comp4_out
tim_ocref_clr4	comp5_out	comp5_out	comp5_out
tim_ocref_clr5	comp6_out	comp6_out	comp6_out

Table 438. Interconnect to the ocref_clr input multiplexer (continued)

Timer OCREF clear signal	Timer OCREF clear signals assignment		
	TIM15	TIM16	TIM17
tim_ocref_clr6	comp7_out	comp7_out	comp7_out
tim_ocref_clr7	comp8_out	comp8_out	comp8_out
tim_ocref_clr[15:8]	Reserved		

38.4.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related autoreload register. The counter can count up, down or both up and down.The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIM_CNT)
- Prescaler register (TIM_PSC)
- Autoreload register (TIM_ARR)
- Repetition counter register (TIM_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIM_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIM_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

Note: An update overrun status bit (UIOVRF in TIM_SR register) indicates that an update request occurred while an update interrupt is pending (update interrupt flag set). This bit is useful when debugging closed loop applications, to indicate that:

- the update interrupt service routine is too long with respect to the interval between consecutive update events

- the update interrupt service routine was served too late with respect to the subsequent update event

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIM_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting one clock cycle after setting the CEN bit in the TIM_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIM_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 764 and *Figure 765* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 764. Counter timing diagram with prescaler division change from 1 to 2

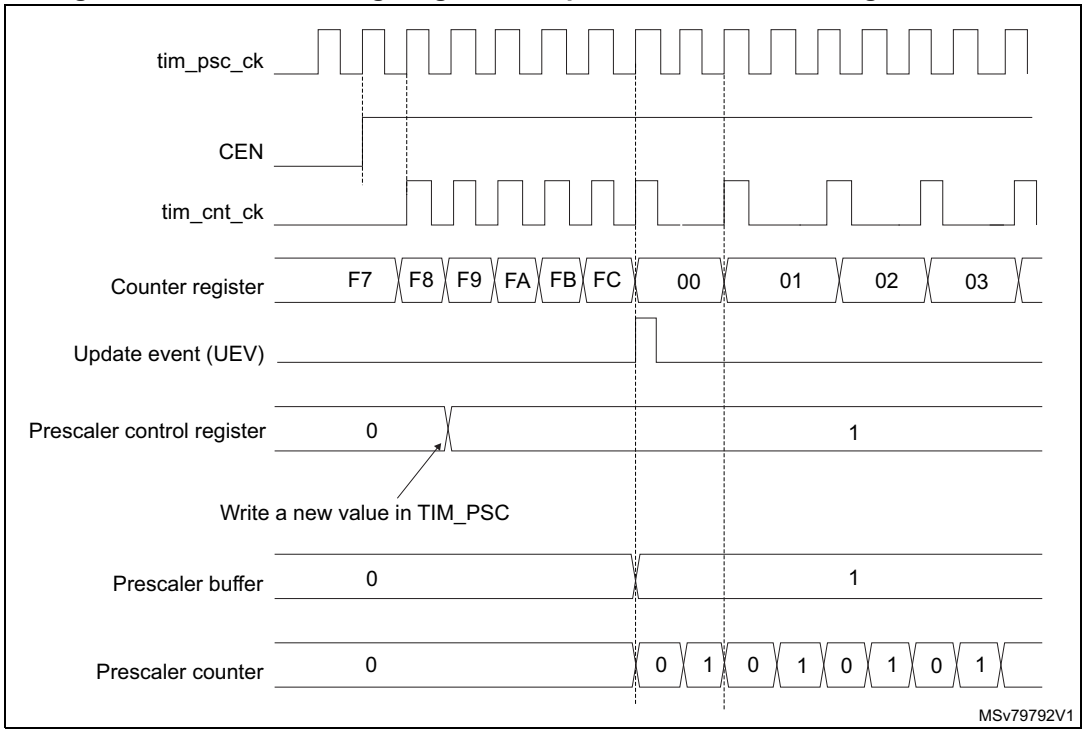
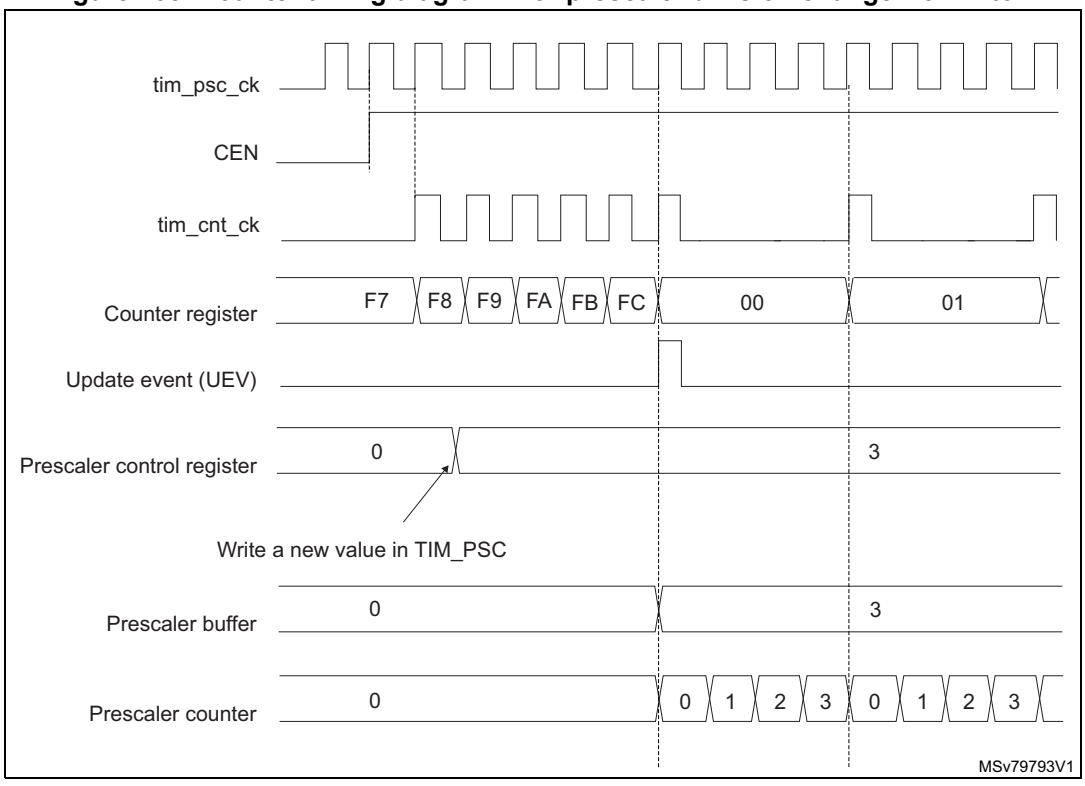


Figure 765. Counter timing diagram with prescaler division change from 1 to 4



38.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIM_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register,
- The autoreload shadow register is updated with the preload value (TIM_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 766. Counter timing diagram, internal clock divided by 1

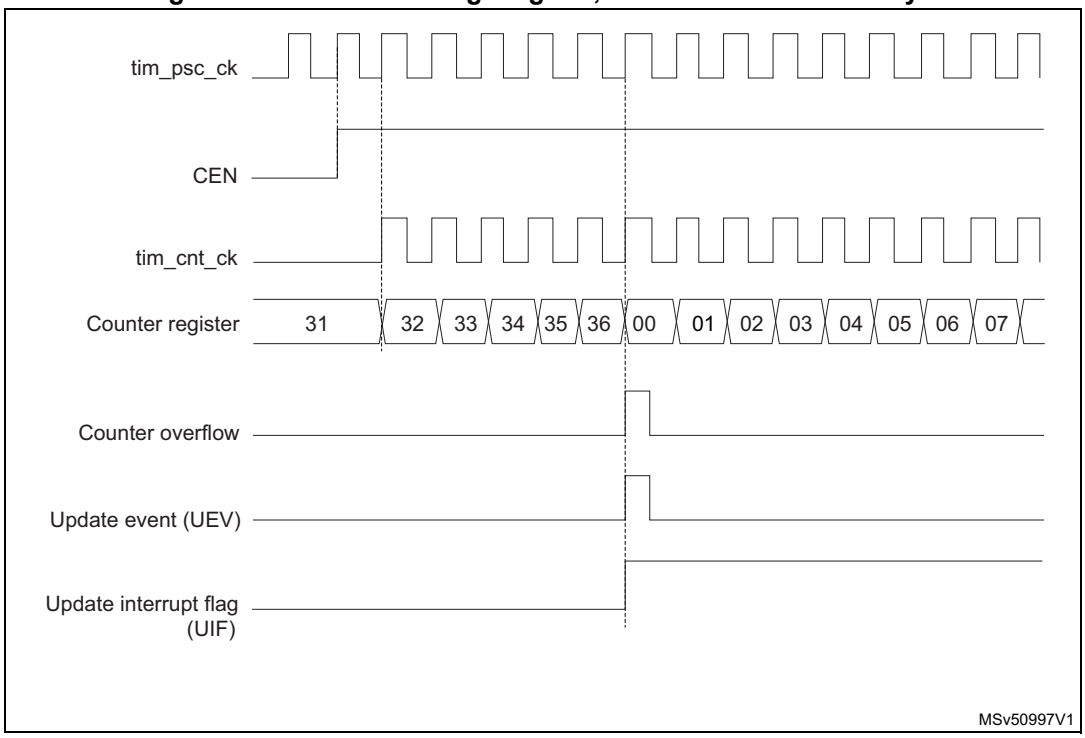


Figure 767. Counter timing diagram, internal clock divided by 2

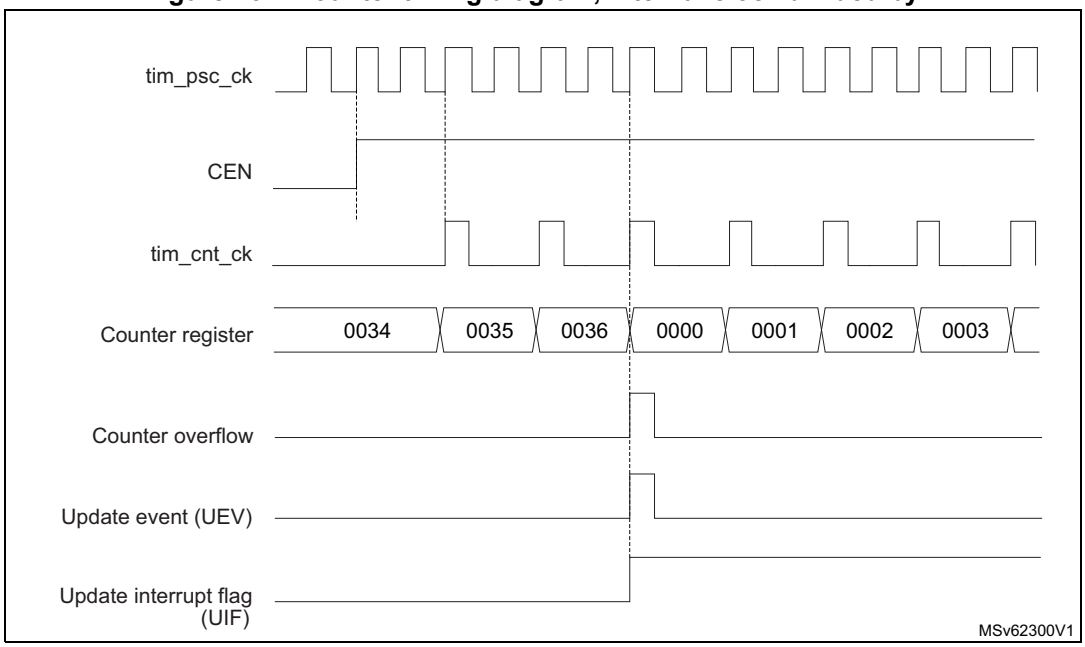


Figure 768. Counter timing diagram, internal clock divided by 4

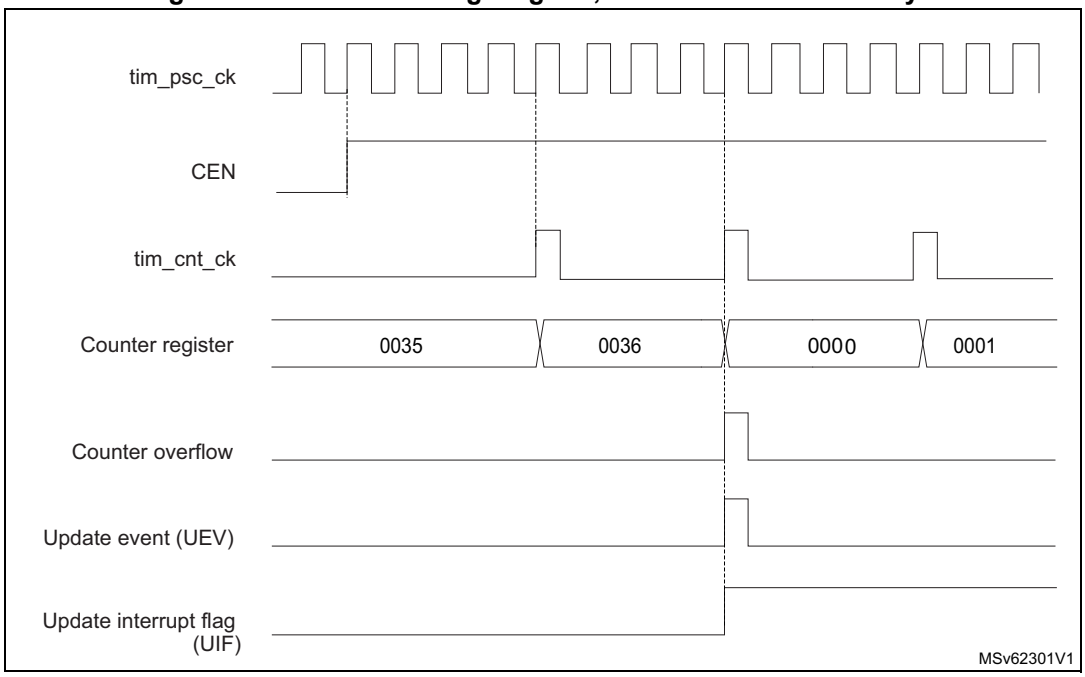


Figure 769. Counter timing diagram, internal clock divided by N

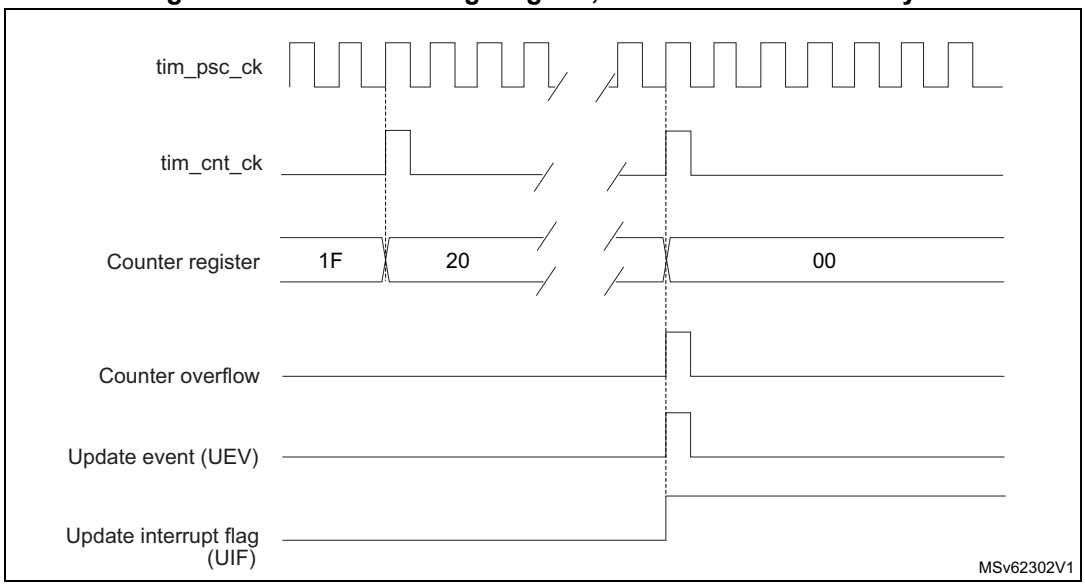


Figure 770. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded)

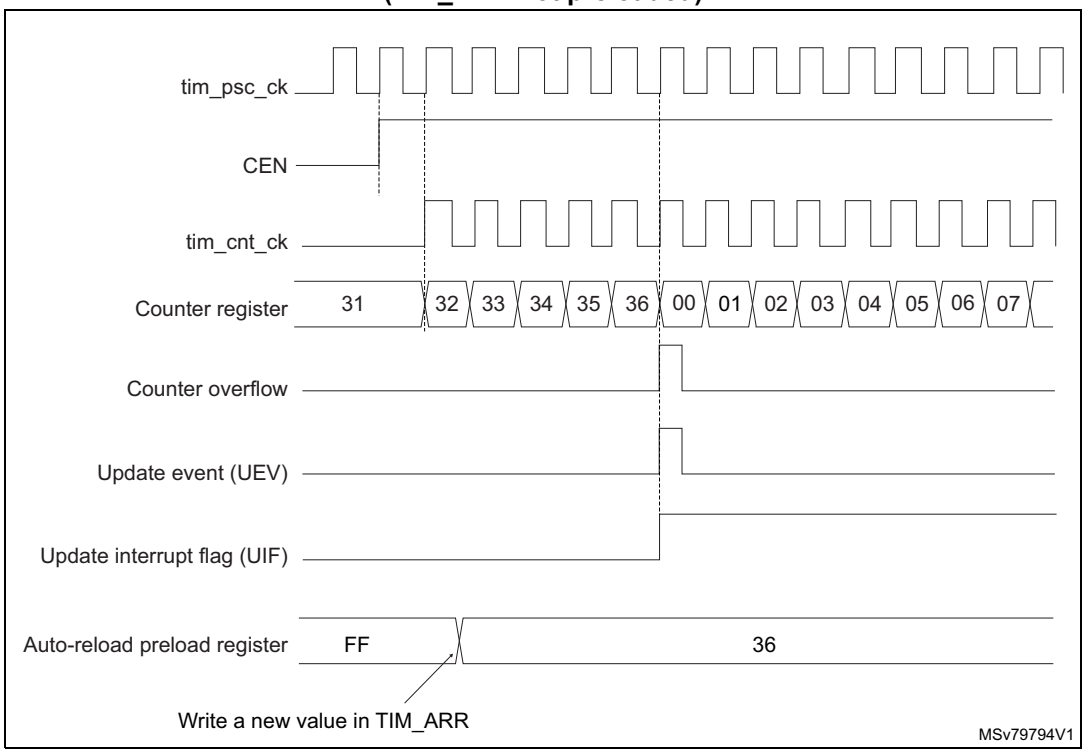
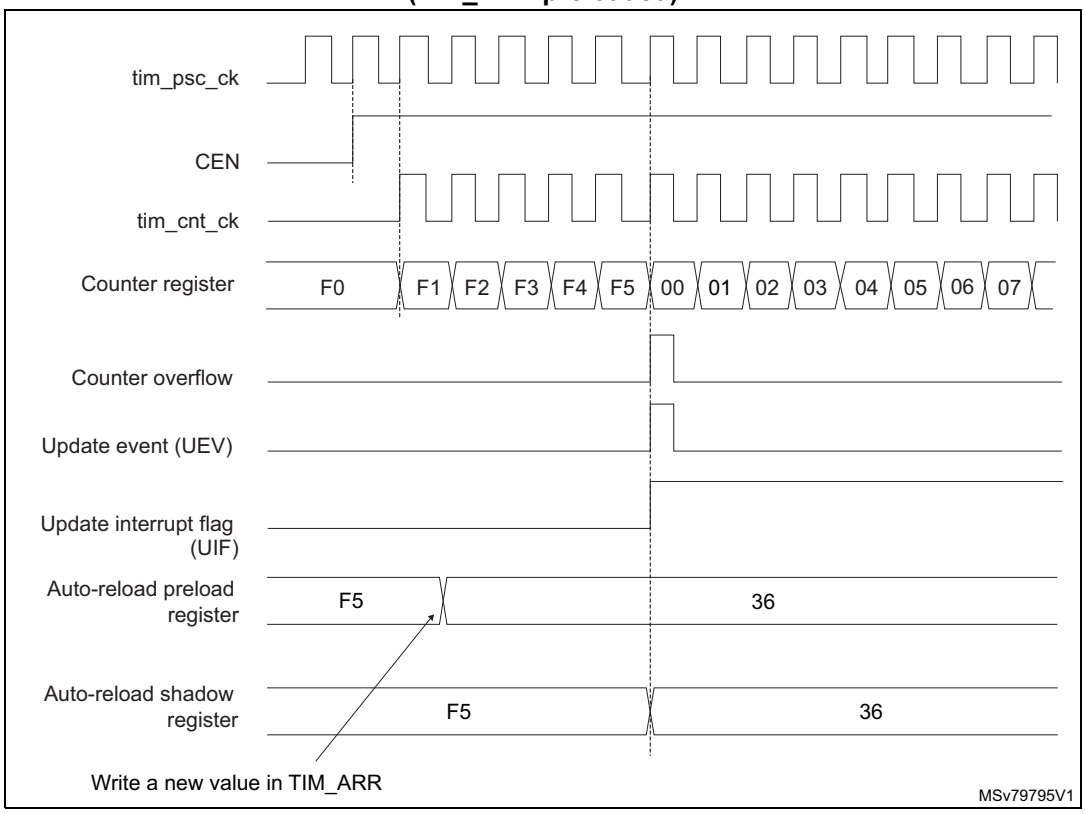


Figure 771. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the autoreload value (content of the TIM_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIM_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 772. Counter timing diagram, internal clock divided by 1

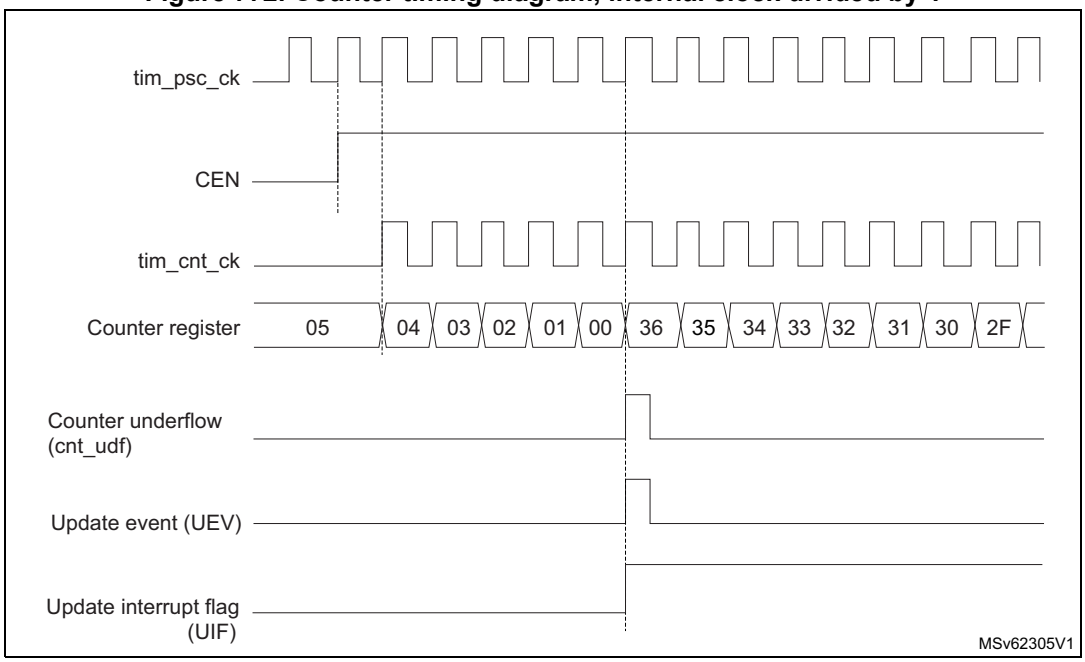


Figure 773. Counter timing diagram, internal clock divided by 2

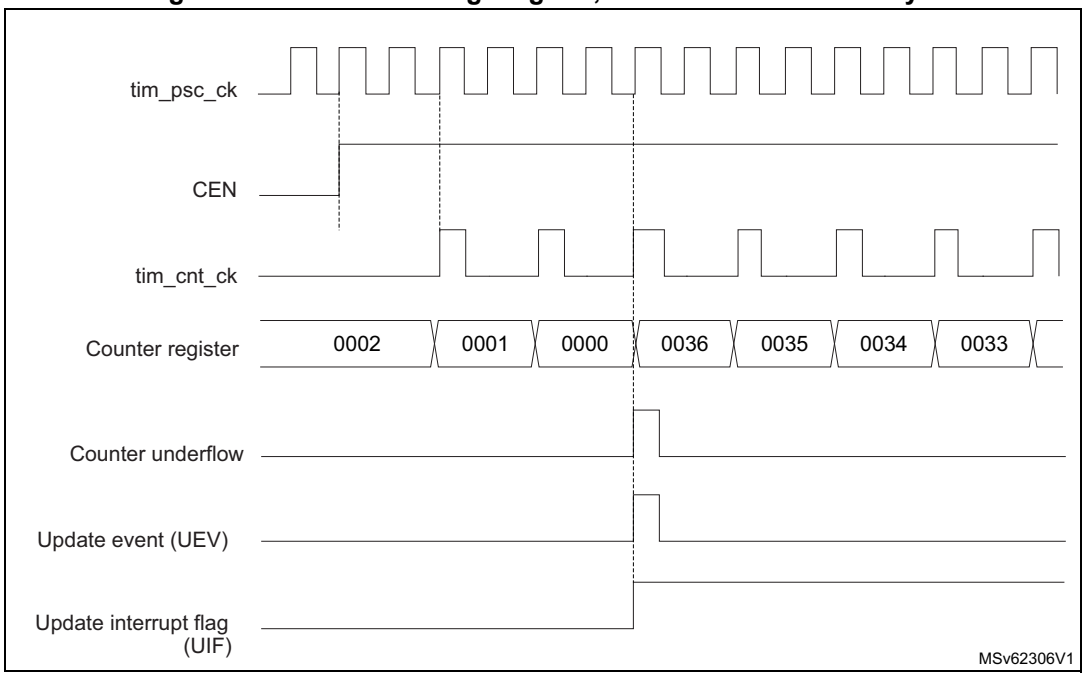


Figure 774. Counter timing diagram, internal clock divided by 4

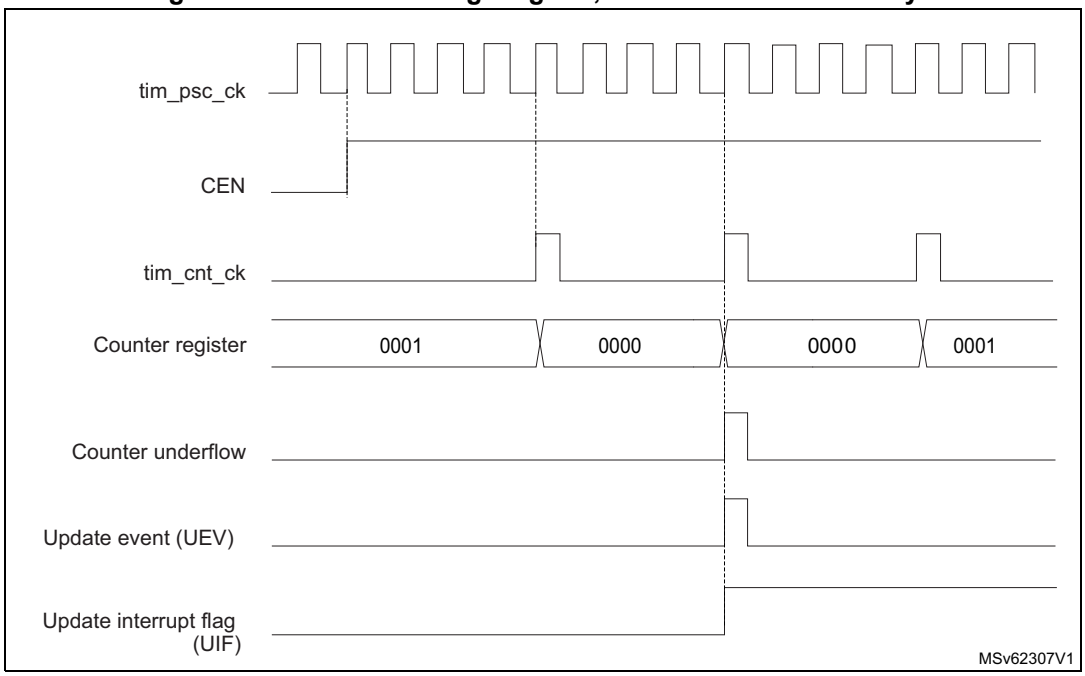


Figure 775. Counter timing diagram, internal clock divided by N

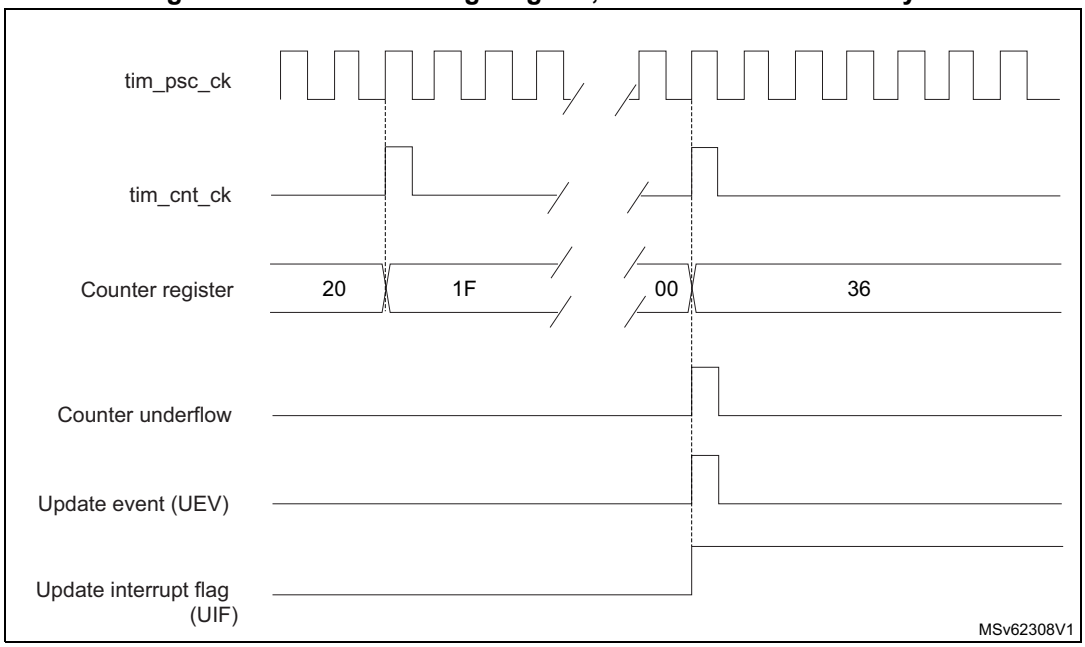
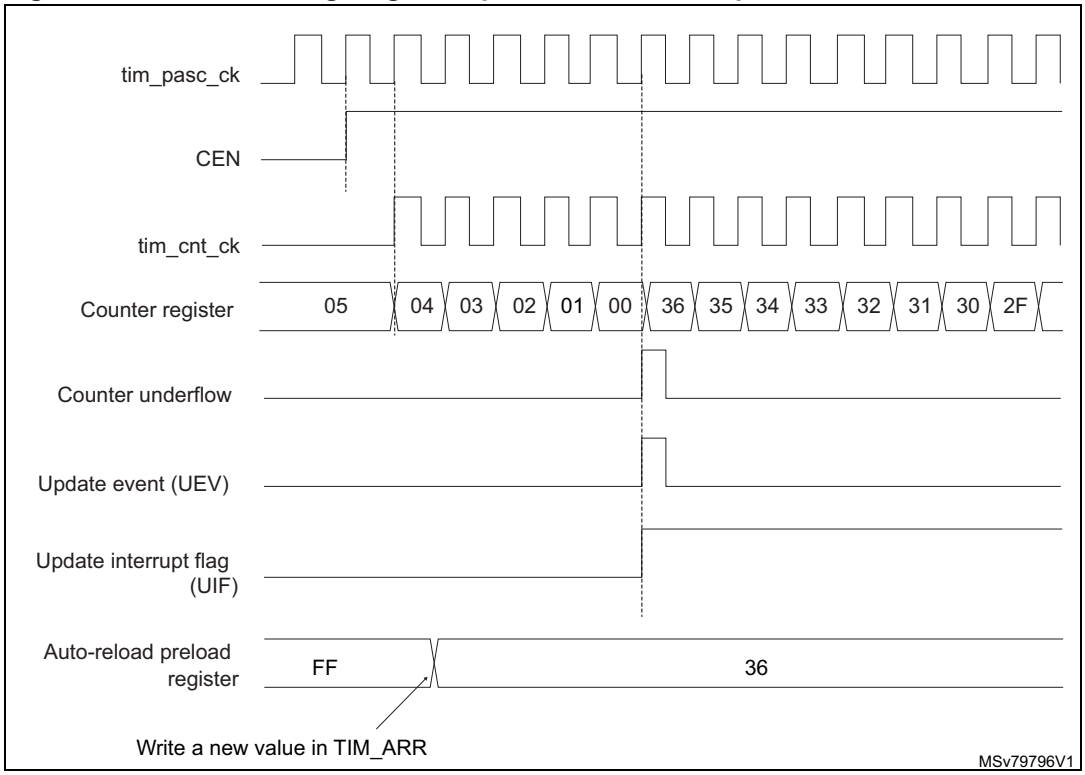


Figure 776. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIM_CR1 register are not equal to 00. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the DIR direction bit in the TIM_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

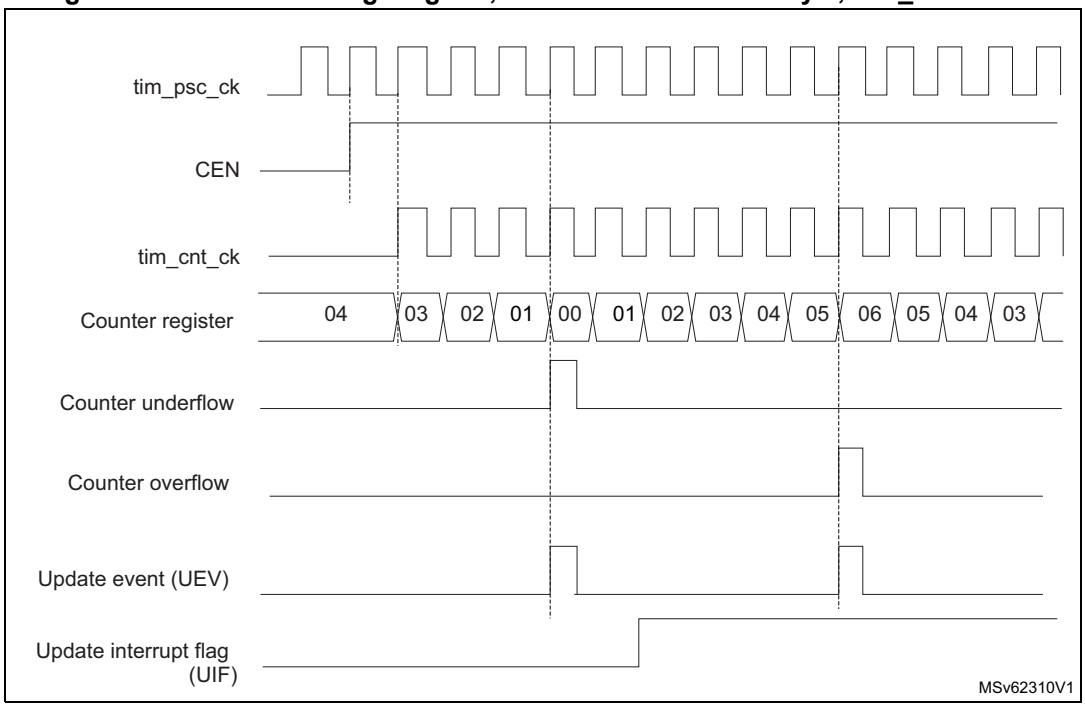
In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register)
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 777. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 34.6: TIM1/TIM8/TIM20 registers*).

Figure 778. Counter timing diagram, internal clock divided by 2

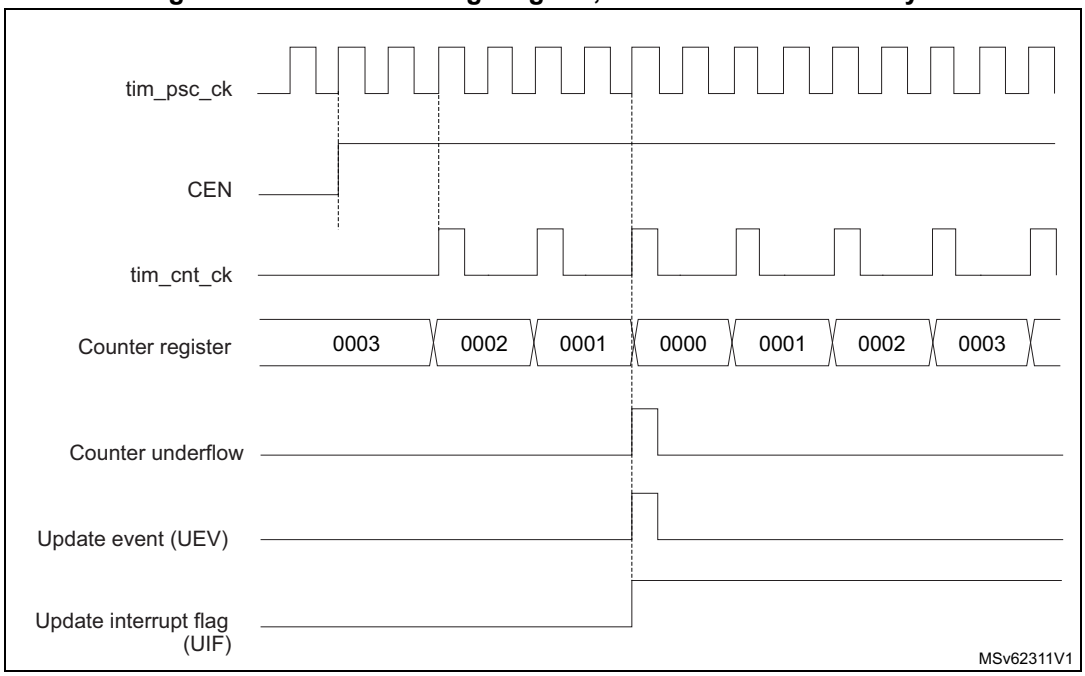


Figure 779. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36

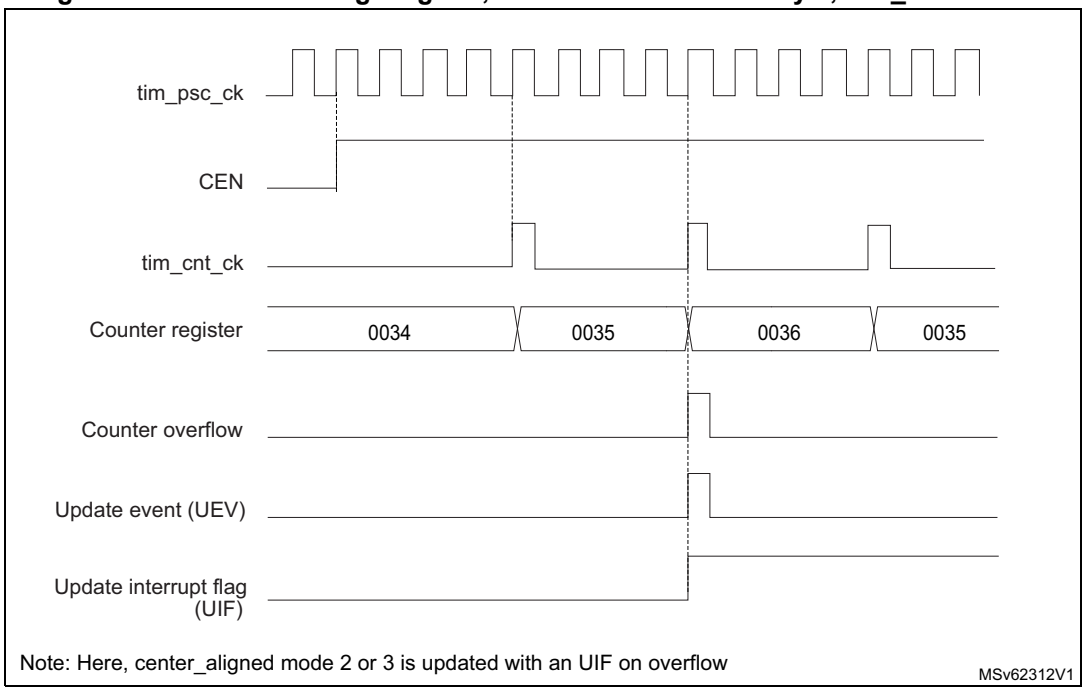


Figure 780. Counter timing diagram, internal clock divided by N

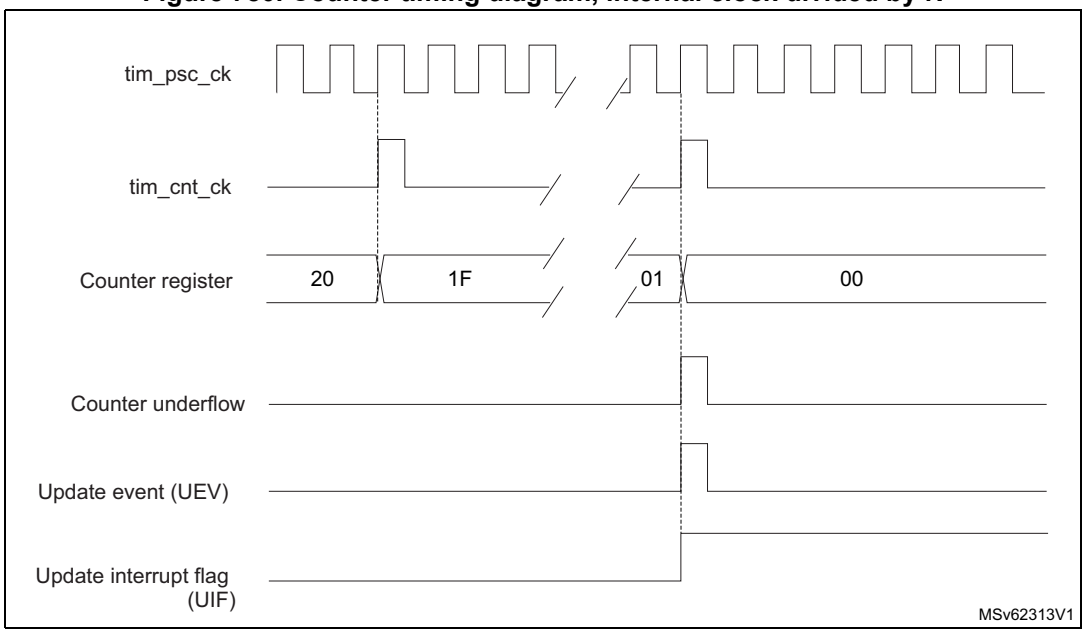


Figure 781. Counter timing diagram, update event with ARPE = 1 (counter underflow)

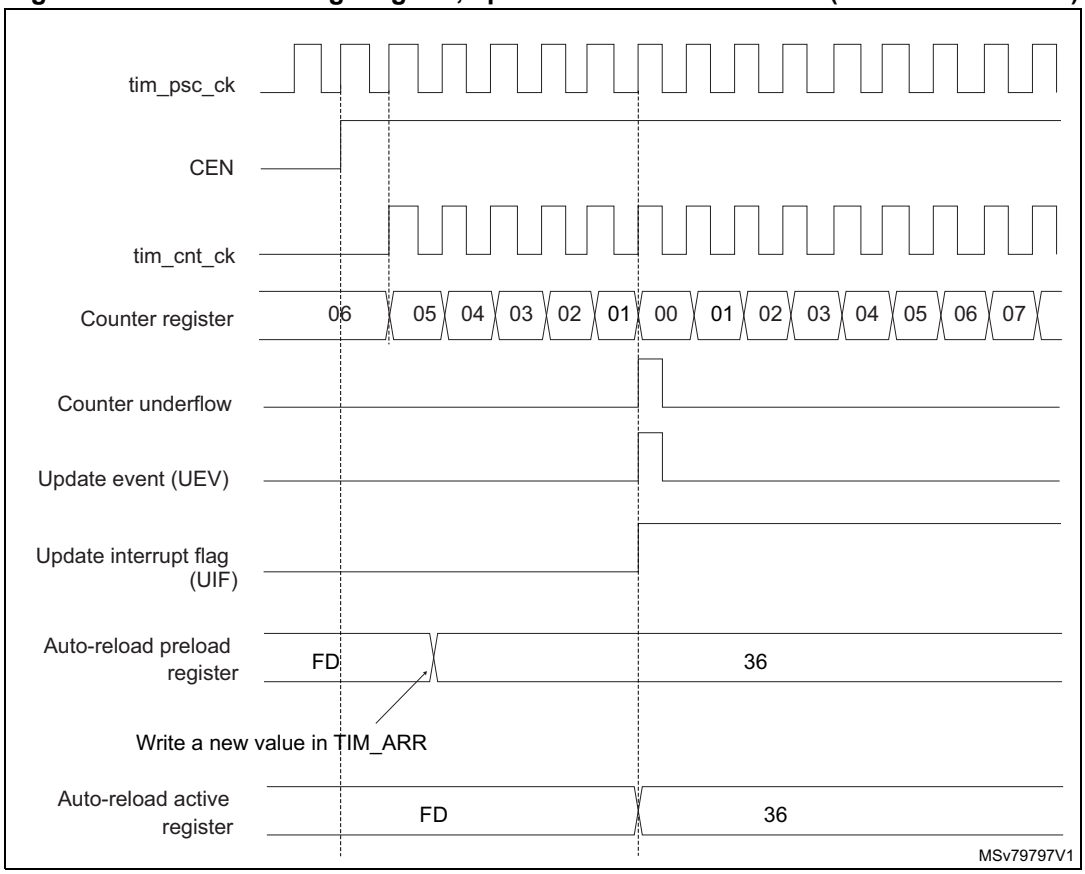
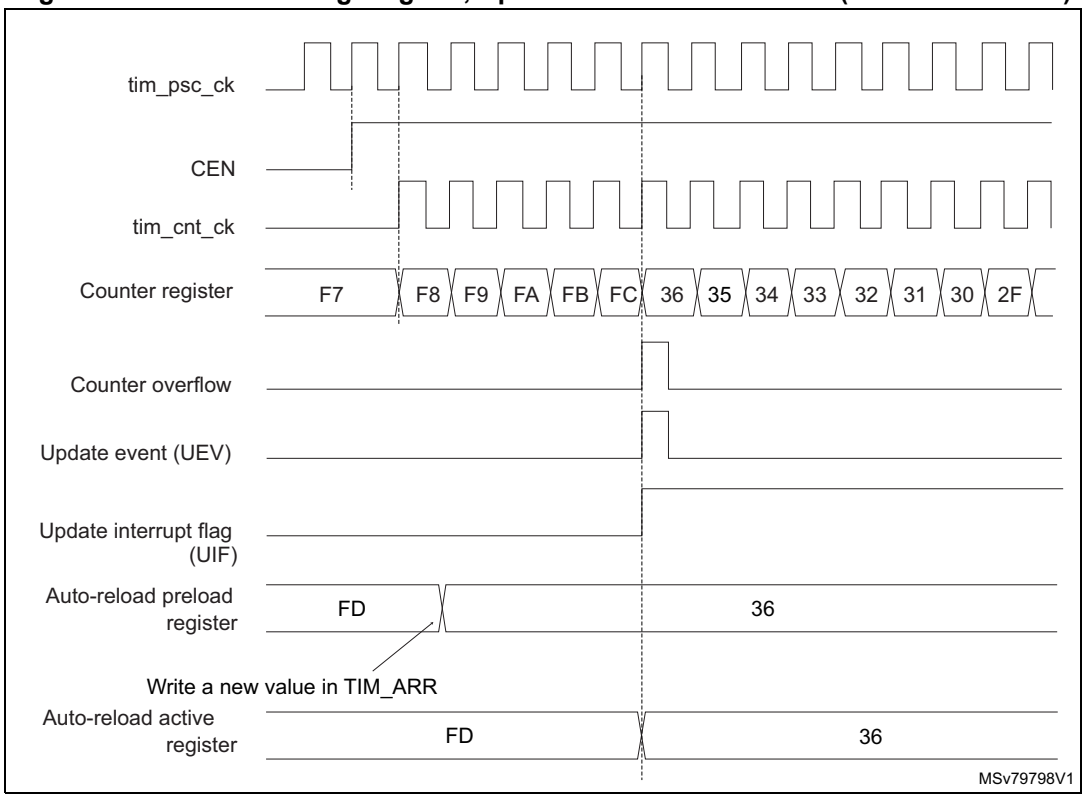


Figure 782. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



38.4.5 Repetition counter

Section 38.4.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIM_ARR autoreload register, TIM_PSC prescaler register, but also TIM_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIM_RCR repetition counter register.

The repetition counter is decremented:

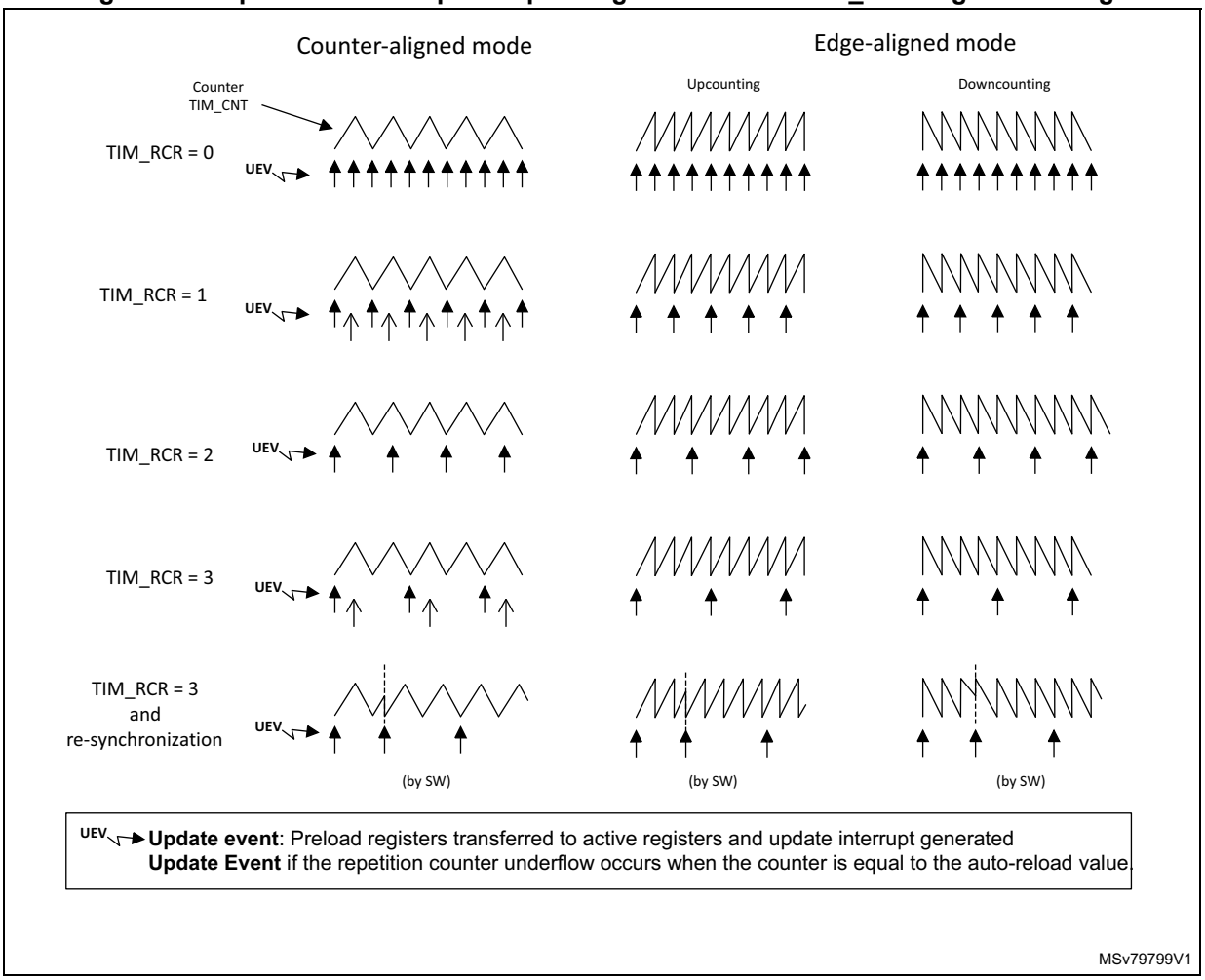
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIM_RCR register value (refer to *Figure 783*). When the update event is generated by software (by setting the UG bit in TIM_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIM_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 783. Update rate examples depending on mode and TIM_RCR register settings



38.4.6 Clock selection

The counter clock can be provided by the following clock sources:

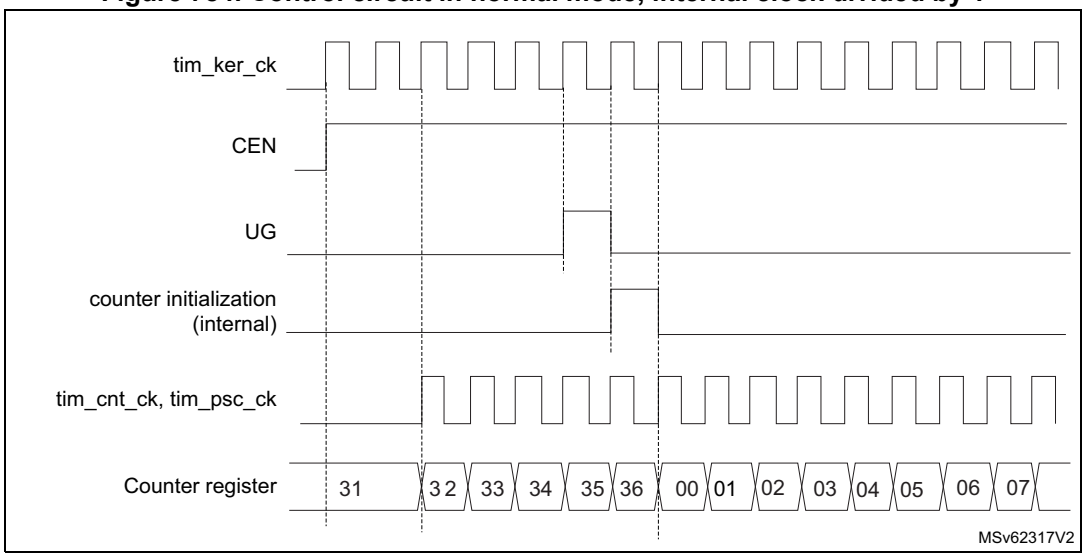
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as a prescaler for TIM15. Refer to *Using one timer to enable another timer* for more details.
- Encoder mode

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 00000), then the CEN (in the TIM_CR1 register) and UG bits (in the TIM_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 784 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

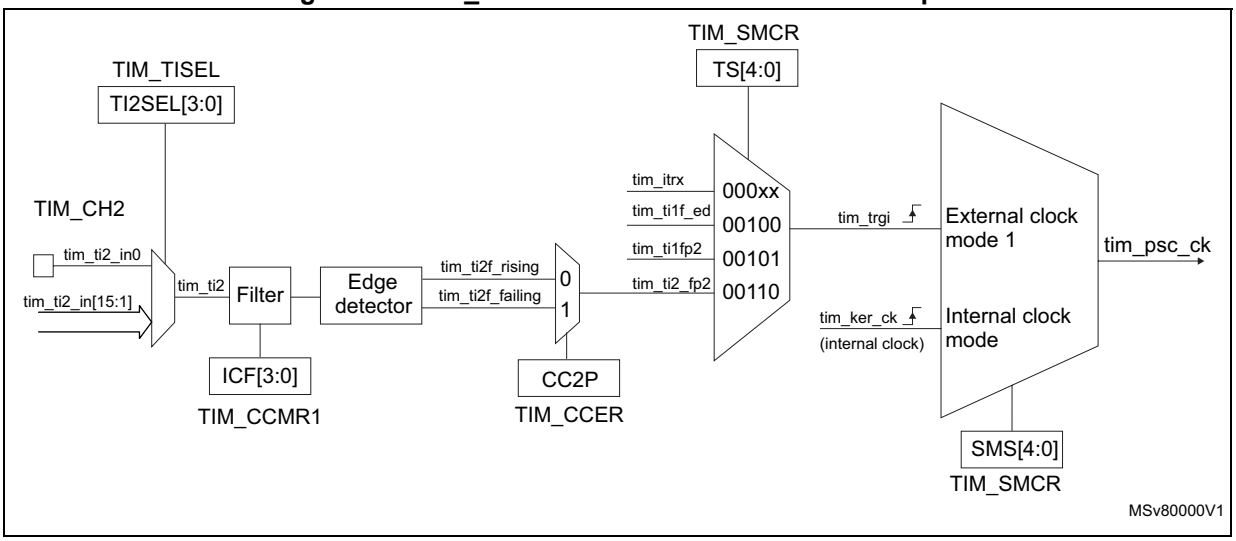
Figure 784. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 00111 in the TIM_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 785. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

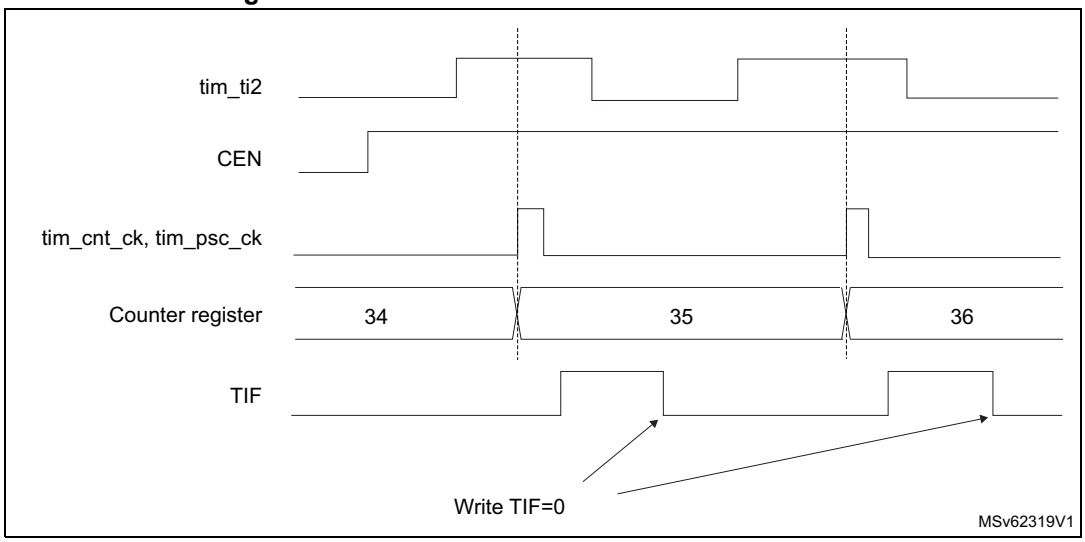
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIM_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIM_CCMR1 register (if no filter is needed, keep IC2F = 0000).
4. Select rising edge polarity by writing CC2P = 0 in the TIM_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 00111 in the TIM_SMCR register.
6. Select tim_ti2 as the trigger input source by writing TS = 00110 in the TIM_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 786. Control circuit in external clock mode 1



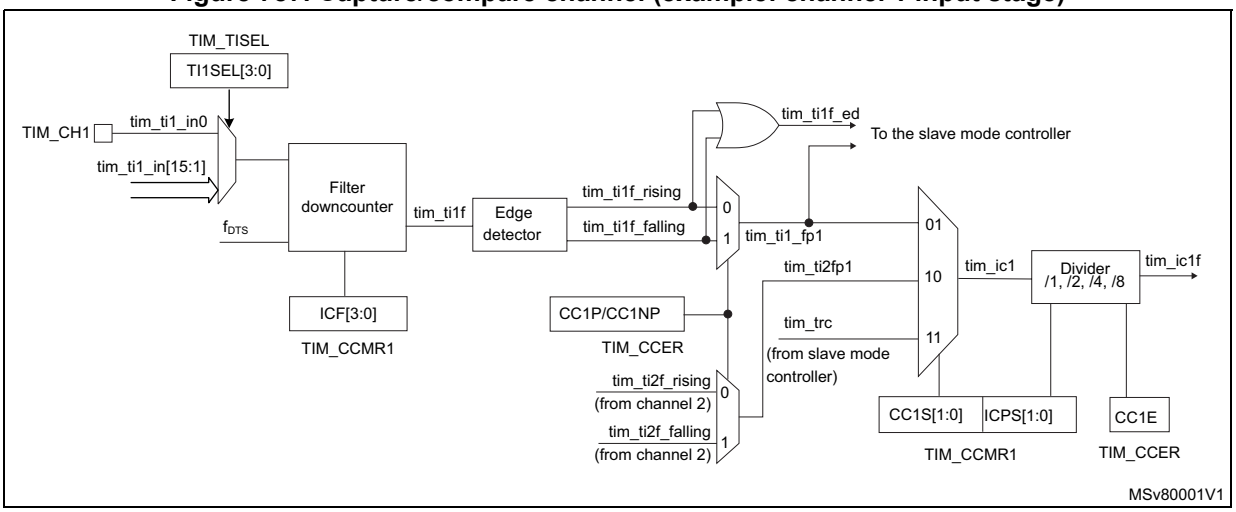
38.4.7 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 787 to *Figure 790* give an overview of one capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 787. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 788. Capture/compare channel 1 main circuit

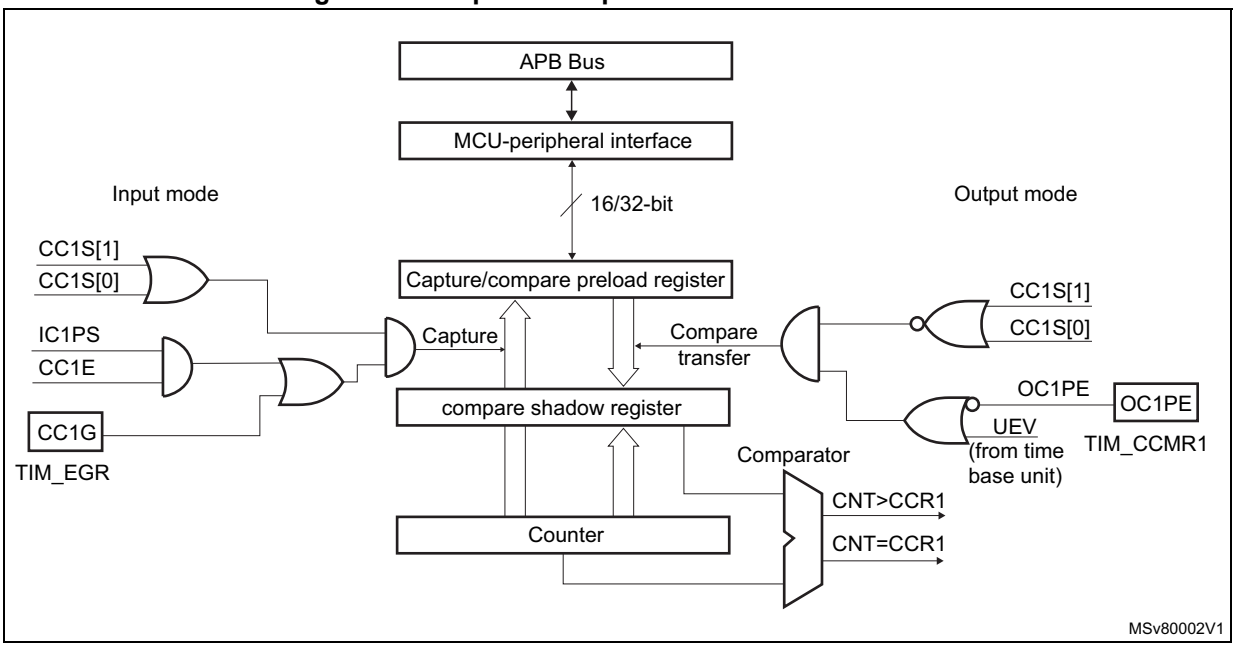


Figure 789. Output stage of capture/compare channel (channel 1)

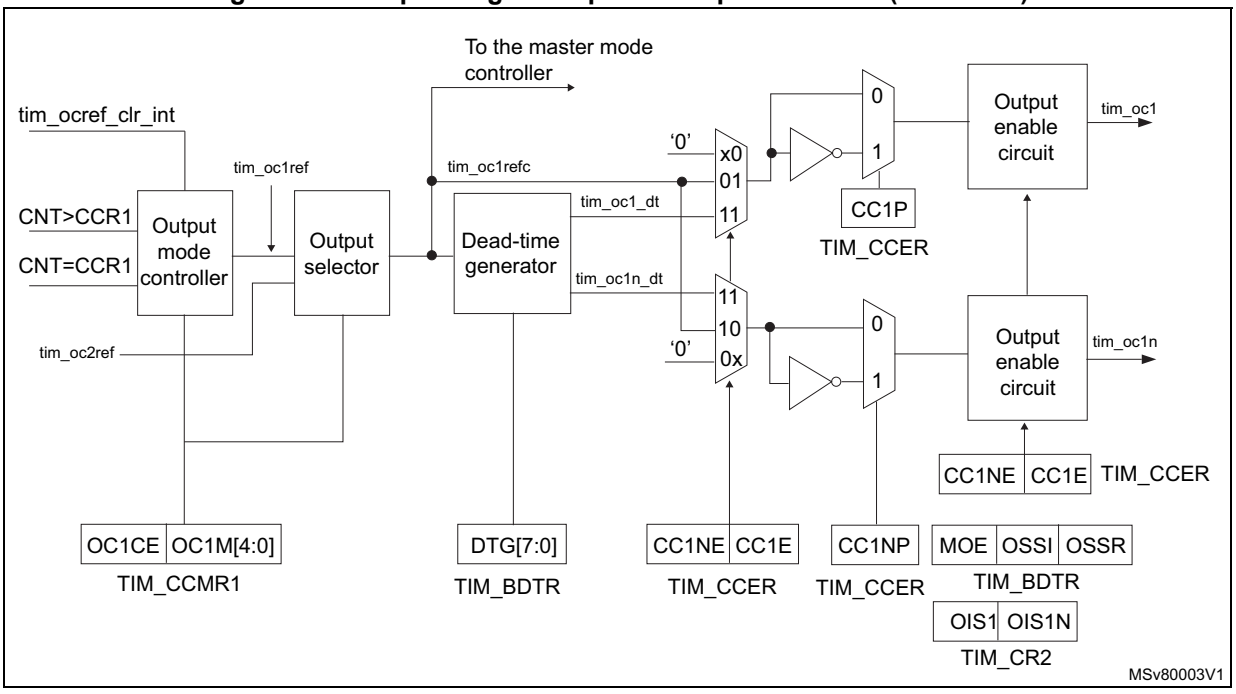
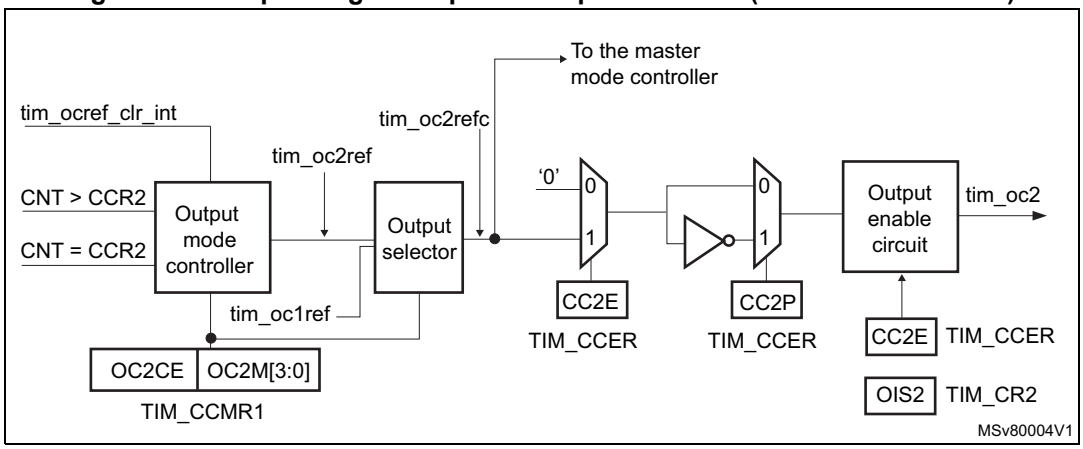


Figure 790. Output stage of capture/compare channel (channel 2 for TIM15)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

38.4.8 Input capture mode

In Input capture mode, the capture/compare registers (TIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIM_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIM_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIM_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIM_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[15:1] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input: TIM_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIM_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input, and the TIM_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIM_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. The user must program a filter duration longer than these five clock cycles. The user can validate a transition on tim_ti1 when eight consecutive samples with the new

level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIM_CCMR1 register.

4. Select the edge of the active transition on the tim_ti1 channel by writing CC1P bit to 0 in the TIM_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the user wants the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIM_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIM_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIM_DIER register, and/or the DMA request by setting the CC1DE bit in the TIM_DIER register.

When an input capture occurs:

- The TIM_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIM_EGR register.

38.4.9 PWM input mode (only for TIM15)

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIM_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges)

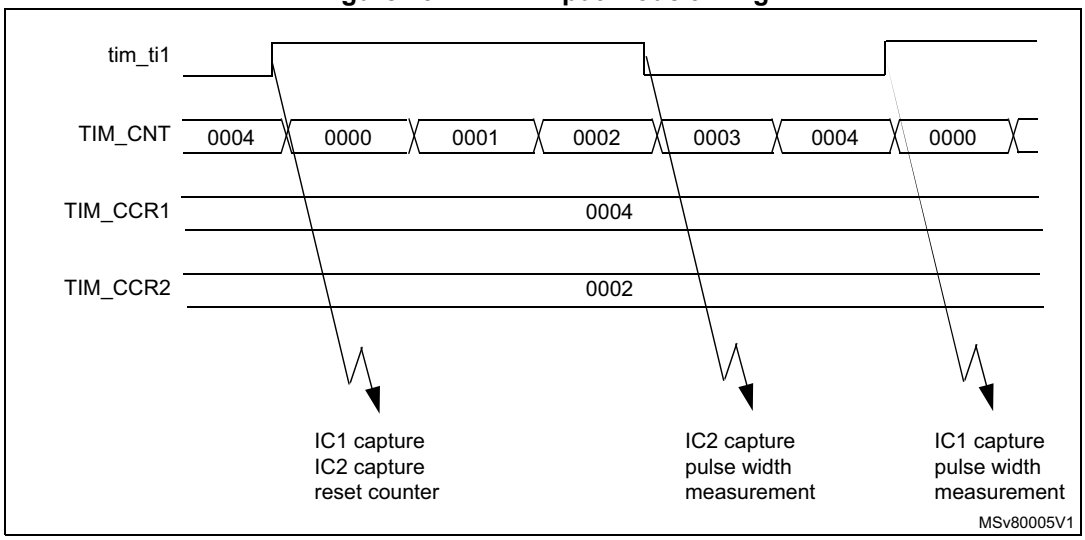
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These two tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIM_CCR1 register) and the duty cycle (in TIM_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input for TIM_CCR1: write the CC1S bits to 01 in the TIM_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIM_CCR1 and counter clear): write the CC1P and CC1NP bits to 0 (active on rising edge).
4. Select the active input for TIM_CCR2: write the CC2S bits to 10 in the TIM_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIM_CCR2): write the CC2P and CC2NP bits to 10 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIM_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 00100 in the TIM_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIM_CCER register.

Figure 791. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

38.4.10 Forced output mode

In output mode (CCxS bits = 00 in the TIM_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 00101 in the OCxM bits in the corresponding TIM_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) → tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 00100 in the TIM_CCMRx register.

Anyway, the comparison between the TIM_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

38.4.11 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIM_CCMRx register) and the output polarity (CCxP bit in the TIM_CCER register). The output pin can keep its level (OCXM = 00000), be set active (OCxM = 00001), be set inactive (OCxM = 00010) or can toggle (OCxM = 00011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIM_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIM_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIM_DIER register, CCDS bit in the TIM_CR2 register for the DMA request selection).

The TIM_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIM_CCMRx register.

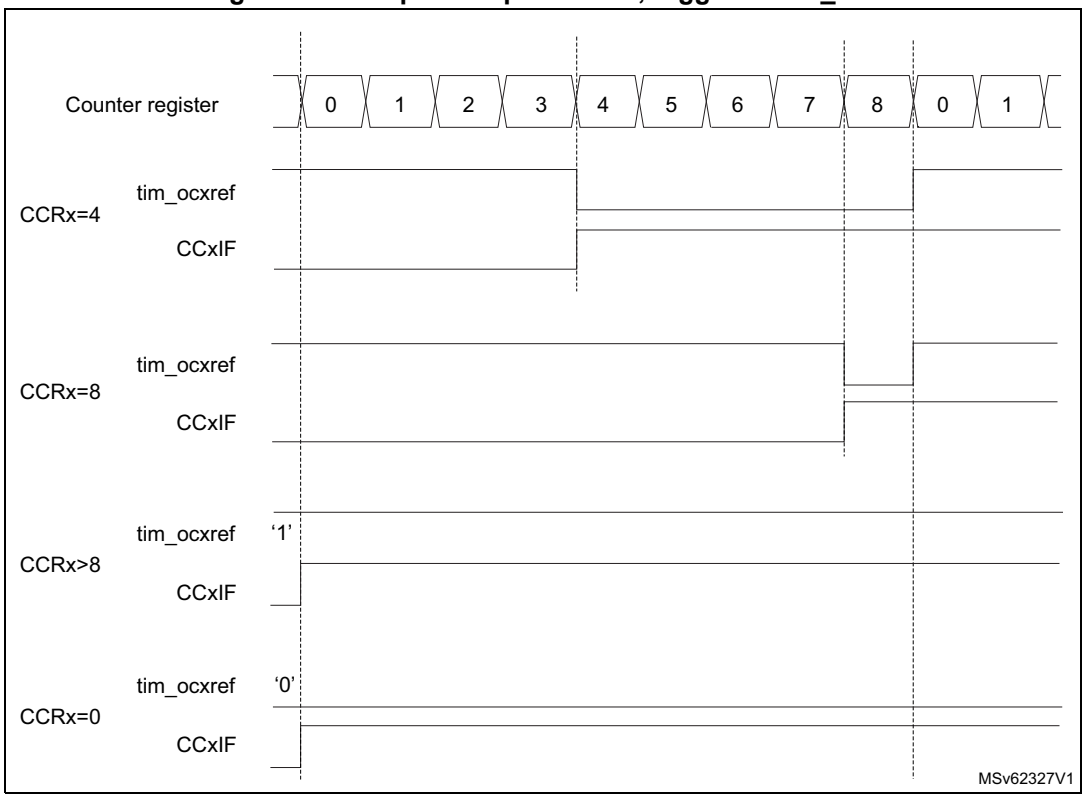
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIM_ARR and TIM_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 00011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIM_CR1 register.

The TIM_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIM_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 792*.

Figure 792. Output compare mode, toggle on tim_oc1



38.4.12 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIM_ARR register and a duty cycle determined by the value of the TIM_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 00110 (PWM mode 1) or 00111 (PWM mode 2) in the OCxM bits in the TIM_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIM_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIM_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIM_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIM_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (TIM_CCER and TIM_BDTR registers). Refer to the TIM_CCER register description for more details.

In PWM mode (1 or 2), TIM_CNT and TIM_CCRx are always compared to determine whether TIM_CCRx ≤ TIM_CNT or TIM_CNT ≤ TIM_CCRx (depending on the direction of the counter).

The TIM15/TIM16/TIM17 are able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIM_CR1 register.

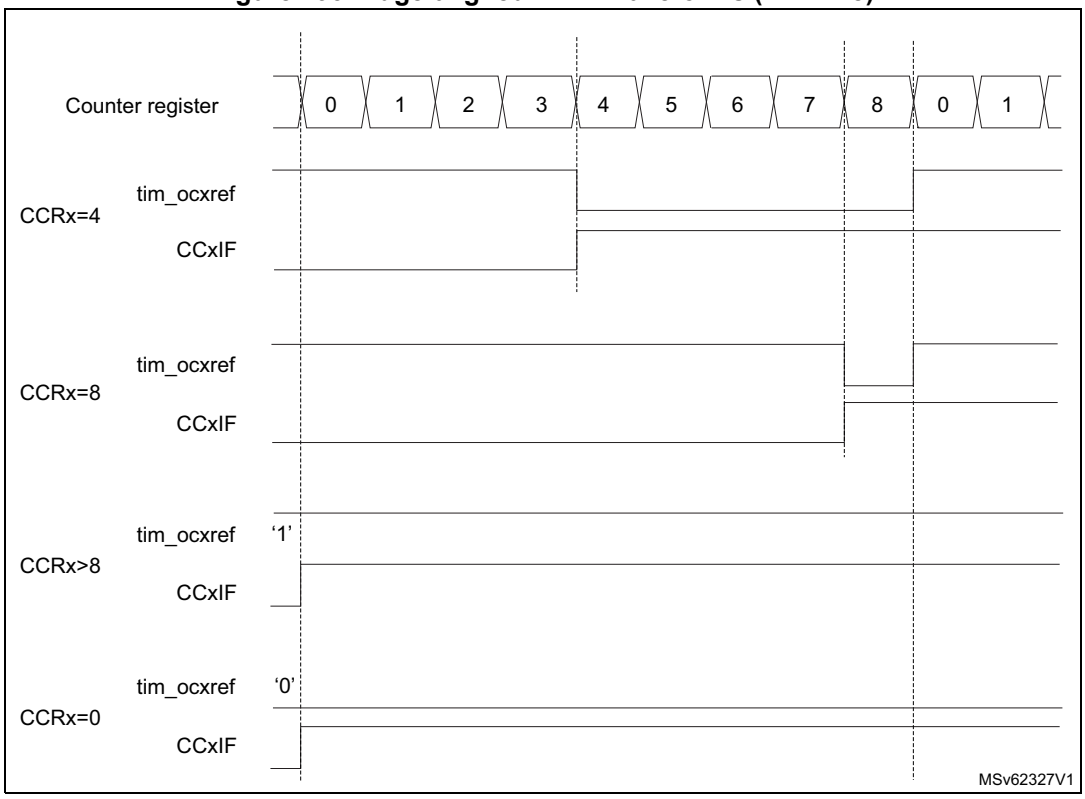
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIM_CR1 register is low. Refer to *Upcounting mode*.

  In the following example, the mode is PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIM_CNT < TIM_CCRx else it becomes low. If the compare value in TIM_CCRx is greater than the autoreload value (in TIM_ARR) then tim_ocxref is held at 1. If the compare value is zero then tim_ocxref is held at 0. *Figure 793* shows some edge-aligned PWM waveforms in an example where TIM_ARR = 8.

Figure 793. Edge-aligned PWM waveforms (ARR = 8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIM_CR1 register is high. Refer to the *Downcounting mode*

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIM_CNT > TIM_CCRx else it becomes high. If the compare value in TIM_CCRx is greater than the autoreload value in TIM_ARR, then tim_ocxref is held at 1. 0% PWM is not possible in this mode.

PWM center-aligned mode

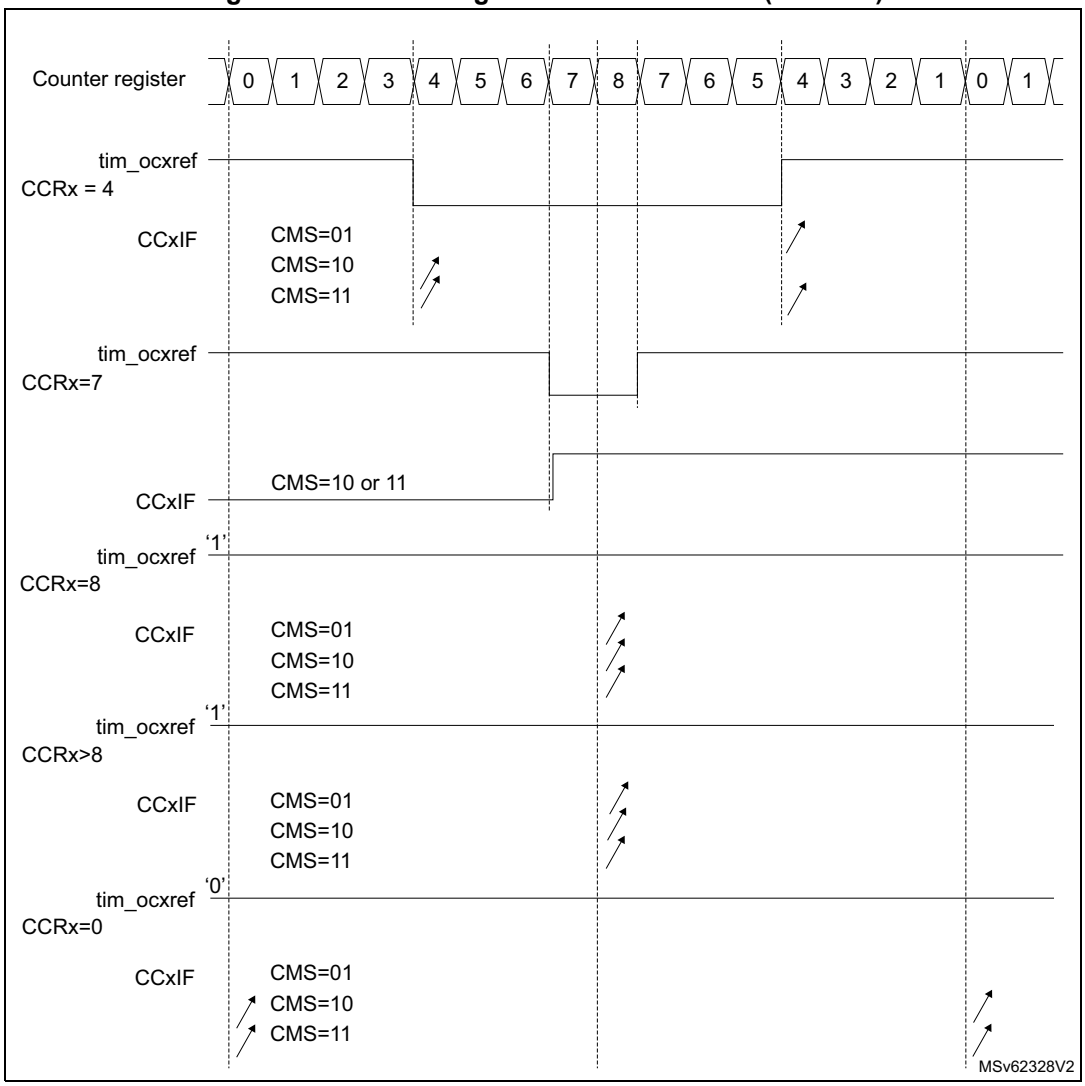
Center-aligned mode is active when the CMS bits in TIM_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the

TIM_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting)*.

Figure 794 shows some center-aligned PWM waveforms in an example where:

- TIM_ARR = 8
- PWM mode is the PWM mode 1
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIM_CR1 register.

Figure 794. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIM_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

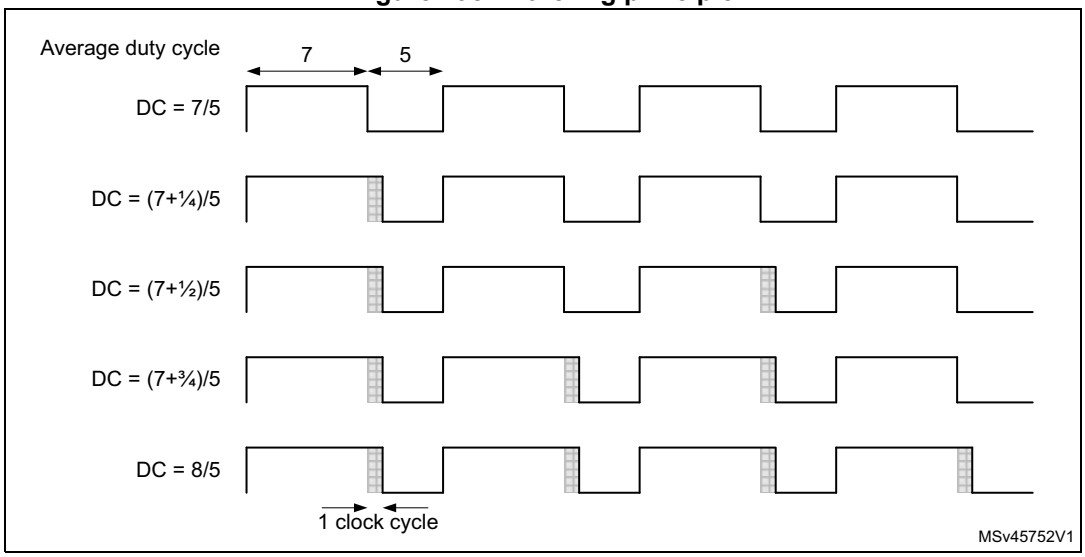
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIM_CNT > TIM_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIM_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIM_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIM_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The *Figure 795* below presents the dithering principle applied to four consecutive PWM cycles.

Figure 795. Dithering principle

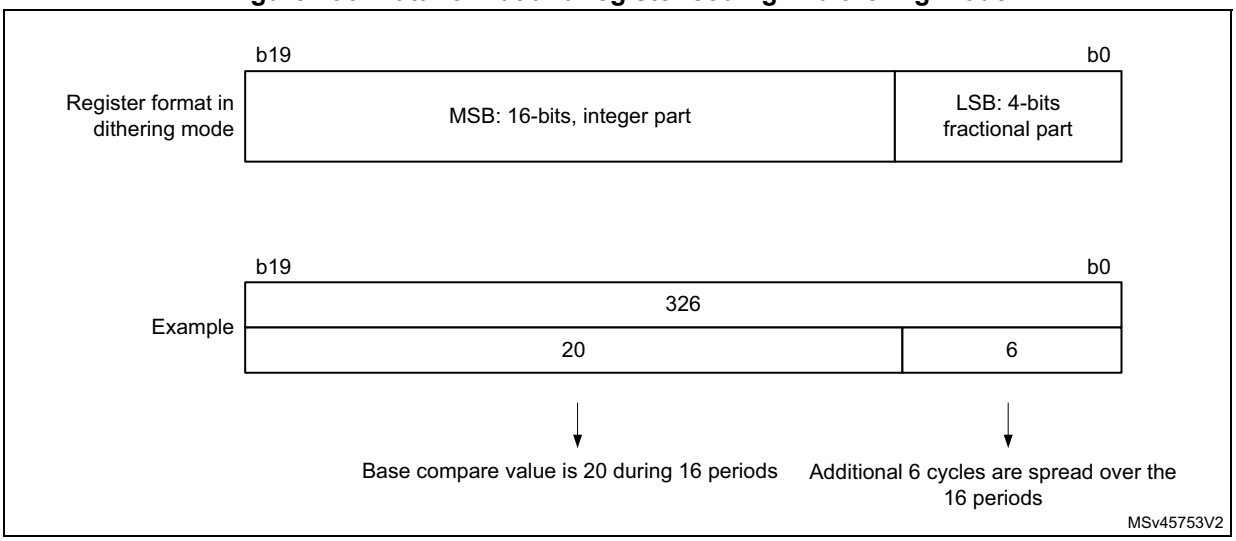


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 796* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 796. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

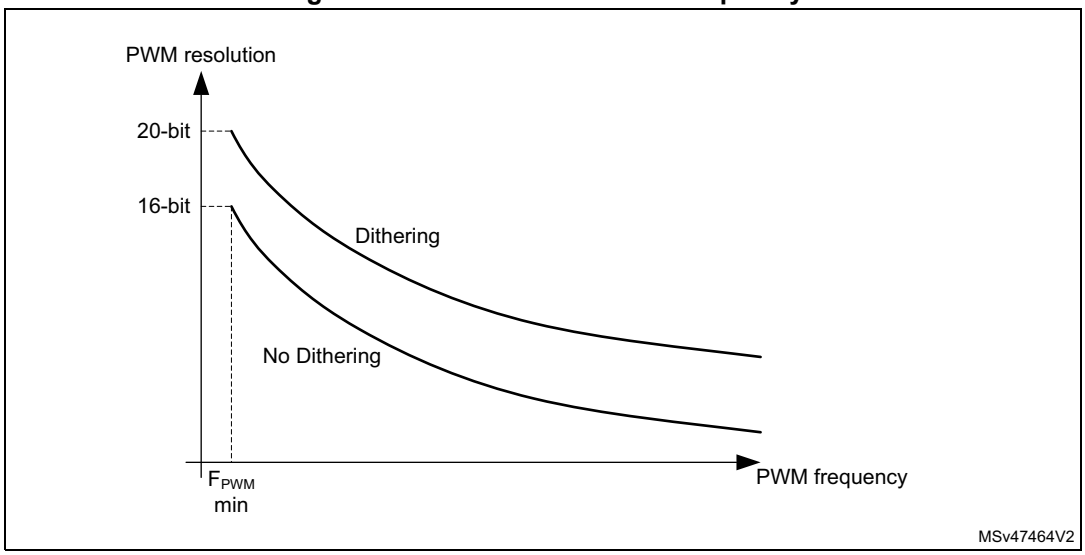
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIM_ARR and TIM_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

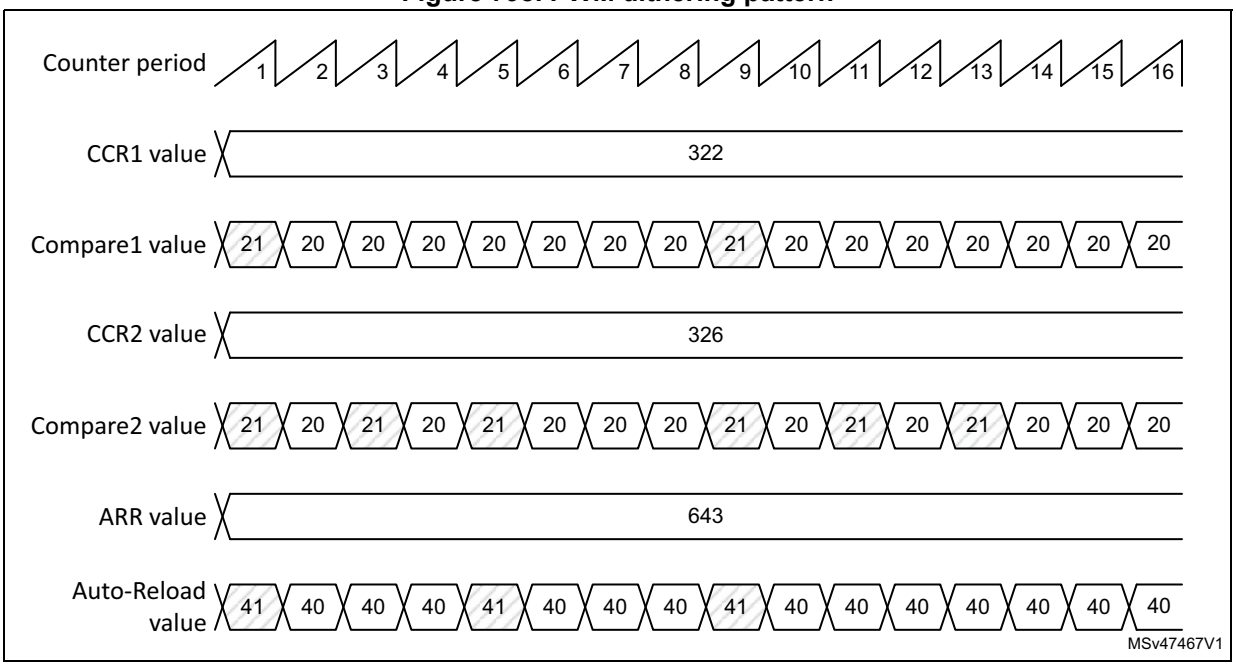
As shown on the *Figure 797* below, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 797. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in the *Figure 798* below.

Figure 798. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in the *Table 439* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 439. CCR and ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIM_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down counting phases as shown in *Figure 799*.

Figure 799. Dithering effect on duty cycle in center-aligned PWM mode

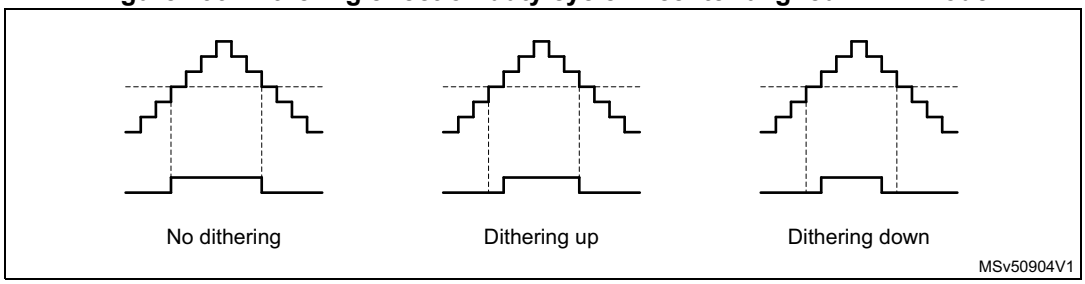


Table 440 shows how the dithering pattern is added in center-aligned PWM mode.

Table 440. CCR register change dithering pattern in center-aligned PWM mode

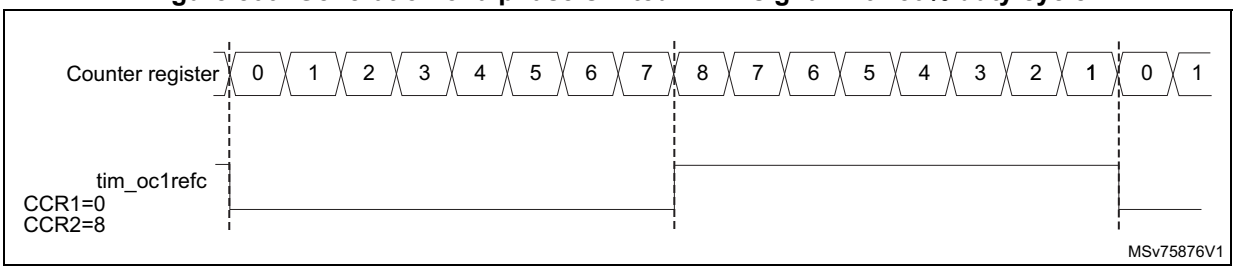
LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

38.4.13 Asymmetric PWM mode (TIM15 only)

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and the phase-shift are determined by a pair of TIM_CCRx registers. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle. For instance, tim_oc1refc is controlled by TIM_CCR1 and TIM_CCR2.

Figure 800 represents an example of signal that can be generated using asymmetric PWM mode (channels 1 and 2 are configured in asymmetric PWM mode 2).

Figure 800. Generation of a phase-shifted PWM signal with 50% duty cycle



Multiple asymmetric PWM modes are available to combine different pairs of TIM_CCRx registers and generate the required waveform, as listed in *Table 441*. The selection is done with the OCxM[4:0] bitfield in TIM_CCMRx registers. For each pair of TIM_CCRx registers, two options are given (for example asymmetric PWM mode 3 and 4), one being in PWM mode 1 (channel active as long as the counter is below the TIM_CCRx value), the other one in PWM mode 2 (channel active when the counter is above the TIM_CCRx value).

For the asymmetric PWM modes 1 and 2, it is mandatory to have the same PWM mode for both channels. For instance, it is mandatory to have OC1M[4:0] = OC2M[4:0] = 01110 for the asymmetric PWM mode 1 on oc1refc.

The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bits are not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if a tim_oc1refc signal is generated on channel 1 (asymmetric PWM mode 1), it is possible to output either the regular tim_oc2ref signal on channel 2 or a tim_oc2refc signal resulting from asymmetric PWM mode 1.

Table 441. Asymmetric PWM modes

Output compare mode	OCxM[4:0]	oc1refc	oc2refc	PWM Mode(1)
Asymmetric PWM mode 1	01110	oc1ref (up-counting)		1
Asymmetric PWM mode 2	01111	oc2ref (down-counting)		2

1. PWM mode 1: channel active as long as the counter is below the TIM_CCRx value.
PWM mode 2: channel active when the counter is above the TIM_CCRx value. Unless explicitly mentioned, the PWM mode must be identical for the 2 channels used for the asymmetric mode.

38.4.14 Combined PWM mode (TIM15 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and delay are determined by the two TIM_CCRx registers. Both signals, tim_oc1refc and tim_oc2refc, are made of an OR or AND logical combination of two reference PWMs controlled respectively by TIM_CCR1 and TIM_CCR2.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 01100 (Combined PWM mode 1) or 01101 (Combined PWM mode 2) in the OCxM bits in the TIM_CCMRx register.

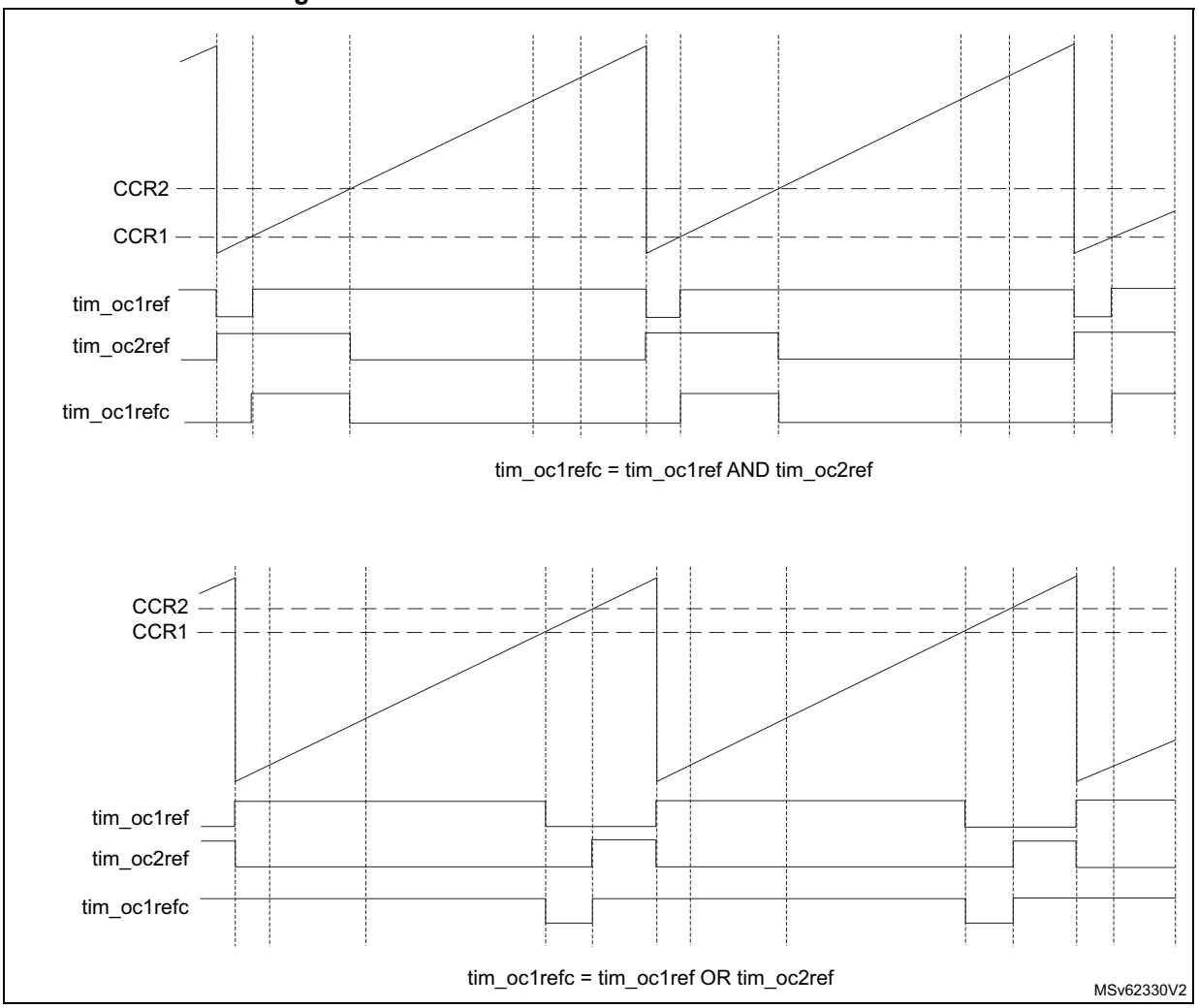
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 801 represents an example of signals that can be generated using Combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.

Figure 801. Combined PWM mode on channel 1 and 2



38.4.15 Complementary outputs and dead-time insertion

The TIM15/TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters and delays due to power switches)

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIM_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIM_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIM_BDTR and TIM_CR2 registers. Refer to *Table 448: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15) on page 2321* and *Table 450: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) on page 2353* for more details. In

particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

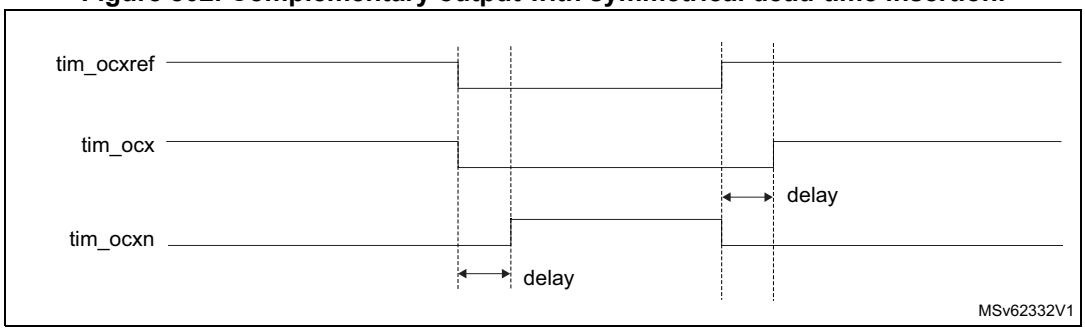
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref. (in these examples CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1)

Figure 802. Complementary output with symmetrical dead-time insertion.



The DTAE bit in the TIM_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 803*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIM_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIM_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set in the TIM_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 803. Asymmetrical deadtime

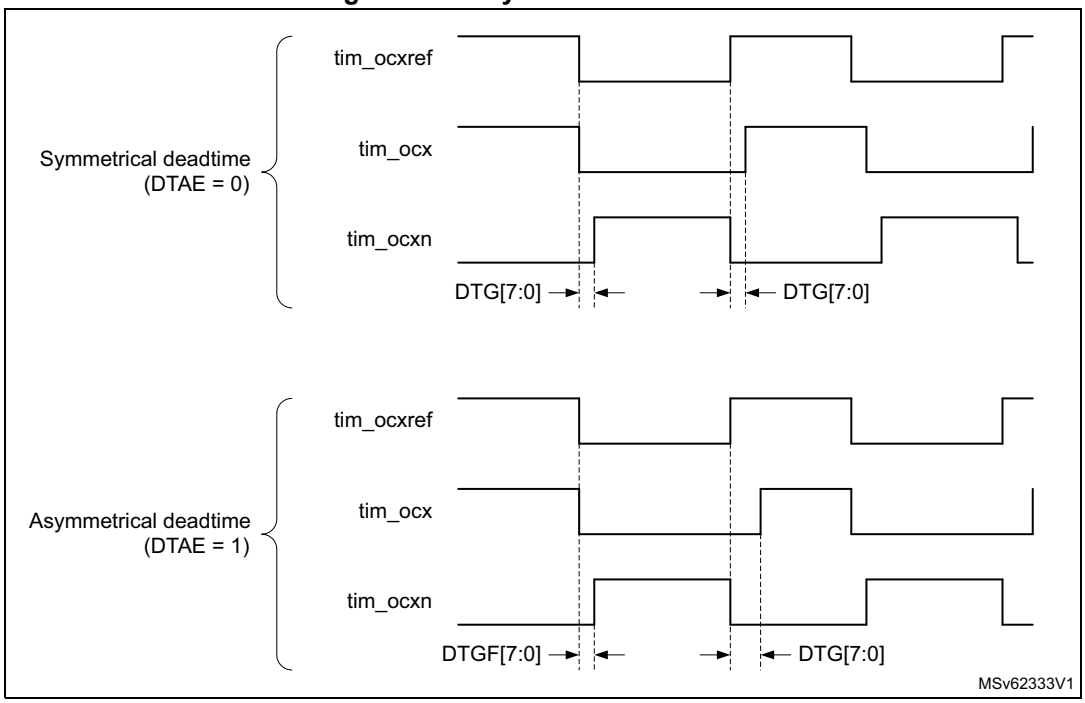


Figure 804. Dead-time waveforms with delay greater than the negative pulse.

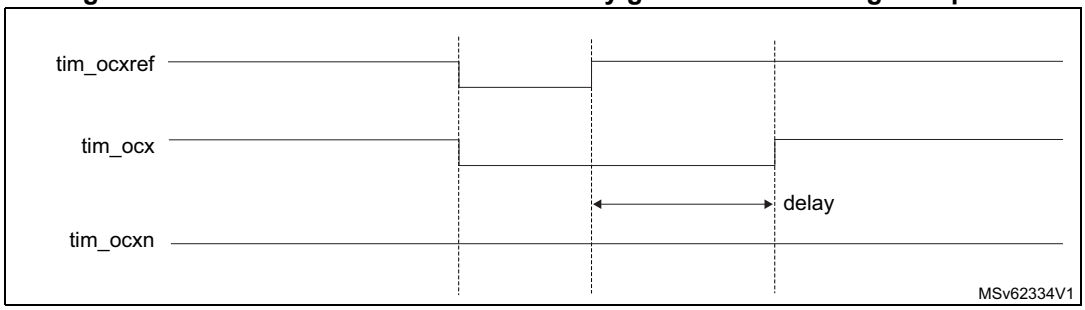
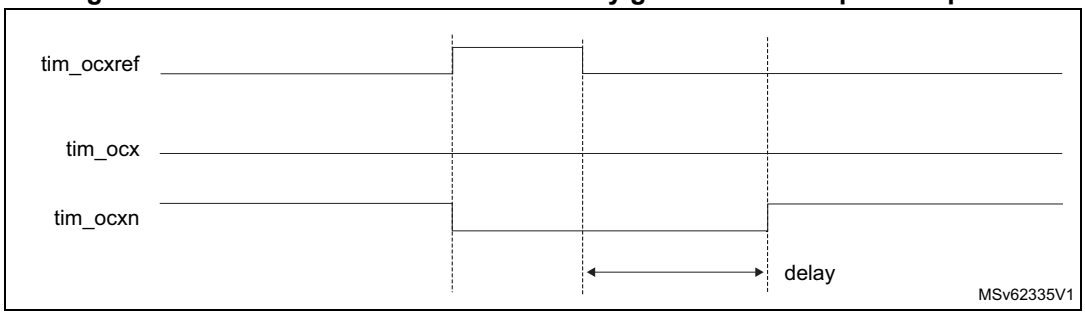


Figure 805. Dead-time waveforms with delay greater than the positive pulse.



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIM_BDTR register. Refer to *Section 38.7.16: TIM break and dead-time register (TIM_BDTR) on page 2325* and *Section 38.8.14: TIM break and dead-time register (TIM_BDTR) on page 2357* for delay calculation.

Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIM_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

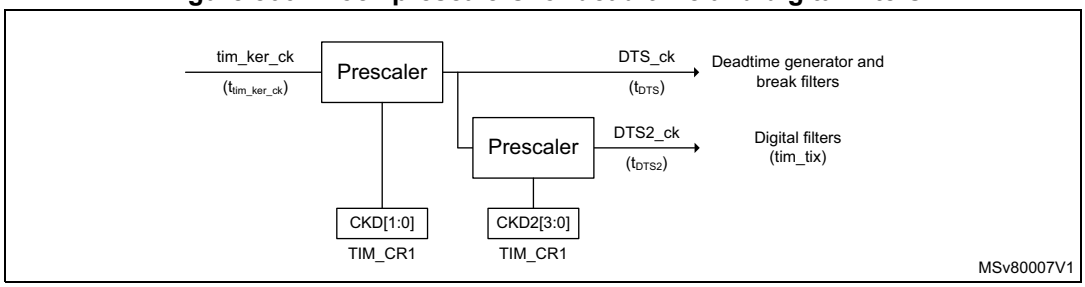
Note: When only tim_ocxn is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP = 0 then tim_ocxn = tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE = CCxNE = 1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

38.4.16 Clock dividers for dead-time generators and digital filters

The dead-time generators and the digital filters (for tim_tix) are using a DTS_ck clock (with a t_{DTS} period), derived from the tim_ker_ck clock.

A prescaler can decrease the frequency to extend the deadtime period programmed with the CKD[1:0] bitfield in the TIM_CR1 register. A second prescaler can decrease the resulting DTS_ck clock to extend solely the digital filters range, with the CKD2[3:0] bitfield in the TIM_CR1 register.

Figure 806. Clock prescalers for dead-time and digital filters



38.4.17 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (such as clock failure, ECC/parity, and errors) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

The output enable signal and output levels during break are depending on several control bits:

- The MOE bit in TIM_BDTR register is used to enable /disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the TIM_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode).
- The OISx and OISxN bits in the TIM_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 448: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15) on page 2321* and *Table 450: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) on page 2353* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIM_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIM_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break is generated by the tim_brk inputs which have:

- Programmable polarity (BKP bit in the TIM_BDTR register).
- Programmable enable bit (BKE bit in the TIM_BDTR register).
- Programmable filter (BKF[3:0] bits in the TIM_BDTR register) to avoid spurious events.

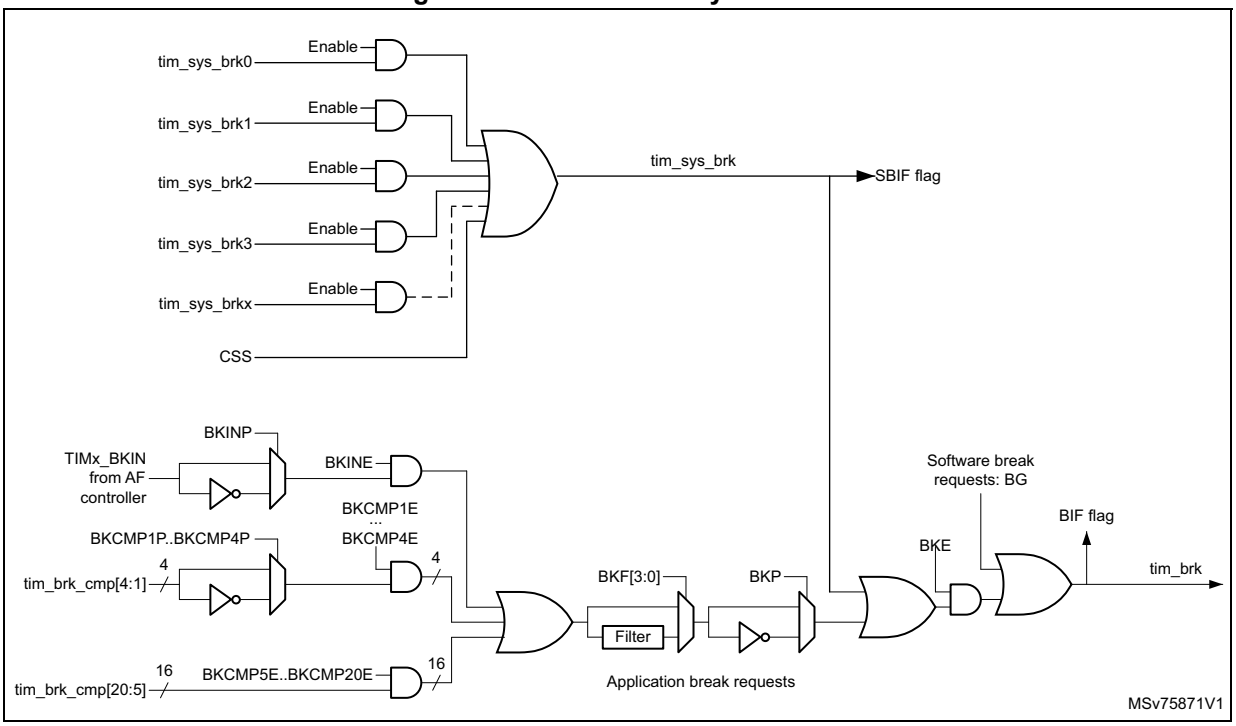
The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIM_AF1 register.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIM_BKIN pins (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering.
- Internal sources:
  - Coming from a tim_brk_cmpx input (refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation).
  - Coming from a system break request on the tim_sys_brk inputs (refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation).

Break events can also be generated by software using BG bit in the TIM_EGR register. All sources are ORed before entering the timer tim_brk inputs, as per *Figure 807* below.

Figure 807. Break circuitry overview



Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail-safe clock mode (for example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIM_CR2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to their active level together. Note that because of the resynchronization on MOE,

the dead-time duration is a bit longer than usual (around 2 tim_ker_ck clock cycles).

– If OSSI = 0 then the timer releases the enable outputs (taken over by the GPIO which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (BIF and BGF bits in the TIM_SR register) is set. An interrupt can be generated if the BIE bit in the TIM_DIER register is set. A DMA request can be sent if the BDE bit in the TIM_DIER register is set.
- If the AOE bit in the TIM_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIM_CR2 register.

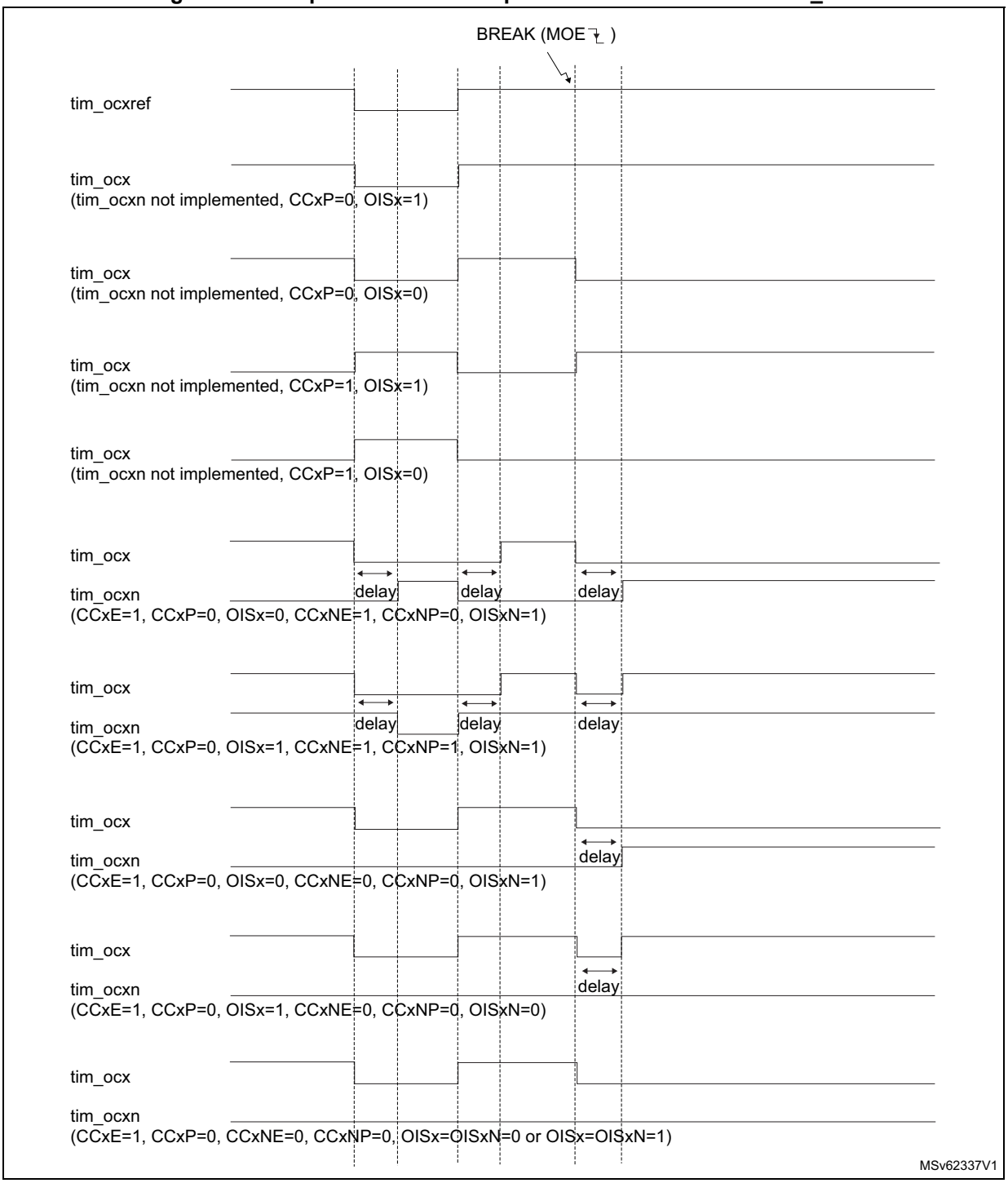
The break inputs are acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the tim_brk input which has a programmable polarity and an enable bit BKE in the TIM_BDTR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable, and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIM_BDTR register. Refer to *Section 38.7.16: TIM break and dead-time register (TIM_BDTR)* and *Section 38.8.14: TIM break and dead-time register (TIM_BDTR)*. The LOCK bits can be written only once after an MCU reset.

The *Figure 808* shows an example of behavior for the outputs in response to a break.

Figure 808. Output behavior in response to a break event on tim_brk



38.4.18 Delayed break

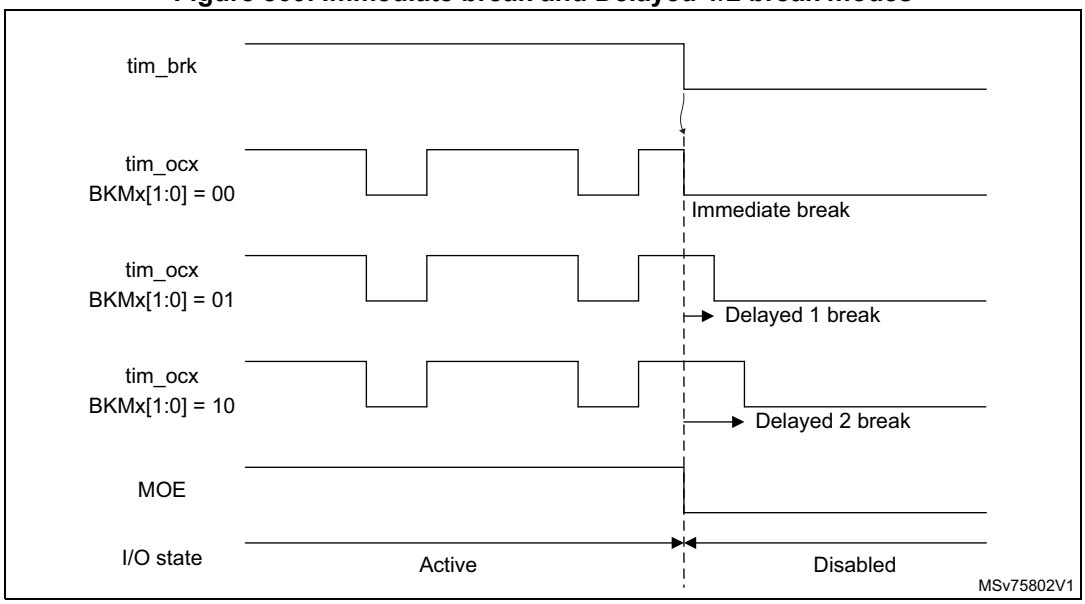
In Neutral-Point Clamped (NPC) multilevel converter topology, a break strategy with immediate outputs shut-down is not suitable, as it may damage the power switches with excessive voltage levels.

The delayed break features provide 2 options delaying the response to a break event, and implement a safe shut-down sequence.

BKMx[1:0] and BKMxN[1:0] bitfields in TIM_MPR1 provides 3 options for each complementary outputs (tim_oc1/tim_oc1n to tim_oc2/tim_oc2n), as show on *Figure 809* below:

- BKM[1:0] = 00: the break response is immediate
- BKM[1:0] = 01: Delayed 1 break. The break response is delayed with a delay defined in the DBK1[7:0] bitfield of the TIM_MPR2 register
- BKM[1:0] = 10: Delayed 2 break. The break response is delayed with a delay defined in the DBK2[7:0] bitfield of the TIM_MPR2 register

Figure 809. Immediate break and Delayed 1/2 break modes



The delayed break mode is triggered when:

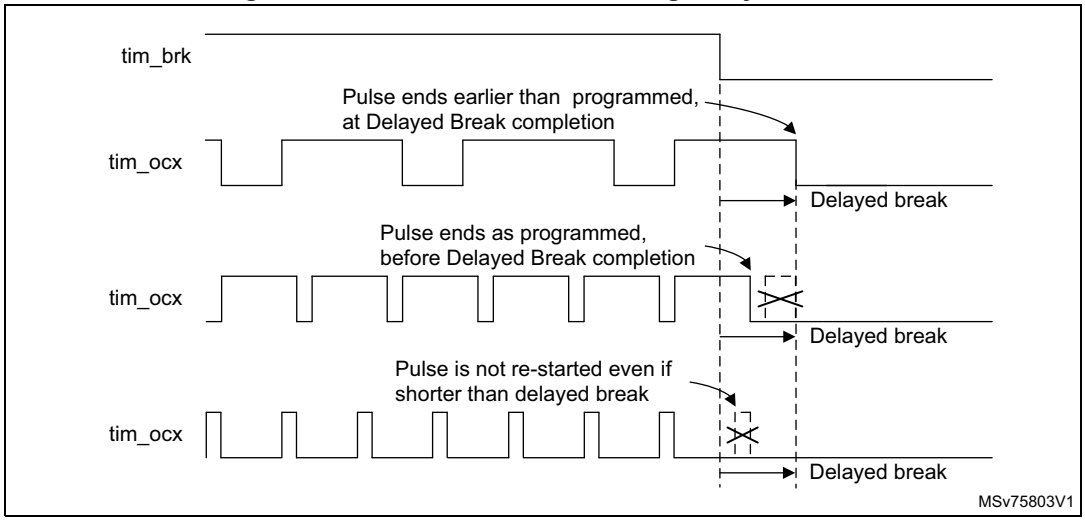
- A break event occurs
- The MOE bit is reset
- When the microcontroller enters debug mode with DBG_TIMx_STOP=1

When delayed break mode is triggered, the MOE bit is immediately cleared, and cannot be set during the delayed break duration. In bidirectional break mode, the break output is reset immediately to indicate the delayed break start to external components. Once the longest of the 2 delays is over, the deadtime generator is enabled to prevent any deadtime violation if OISx/OISxN bits are set.

If the tim_ocx/tim_ocxn signal is reset before the end of the delayed break (due to a compare match in PWM mode, typically), the signal is not maintained during the full delayed break period, to maintain the duty cycle, as shown on *Figure 810* below. Still, the Delayed

break circuitry guarantees that the tim_ocx/tim_ocxn signals cannot be set again once the break event occurred.

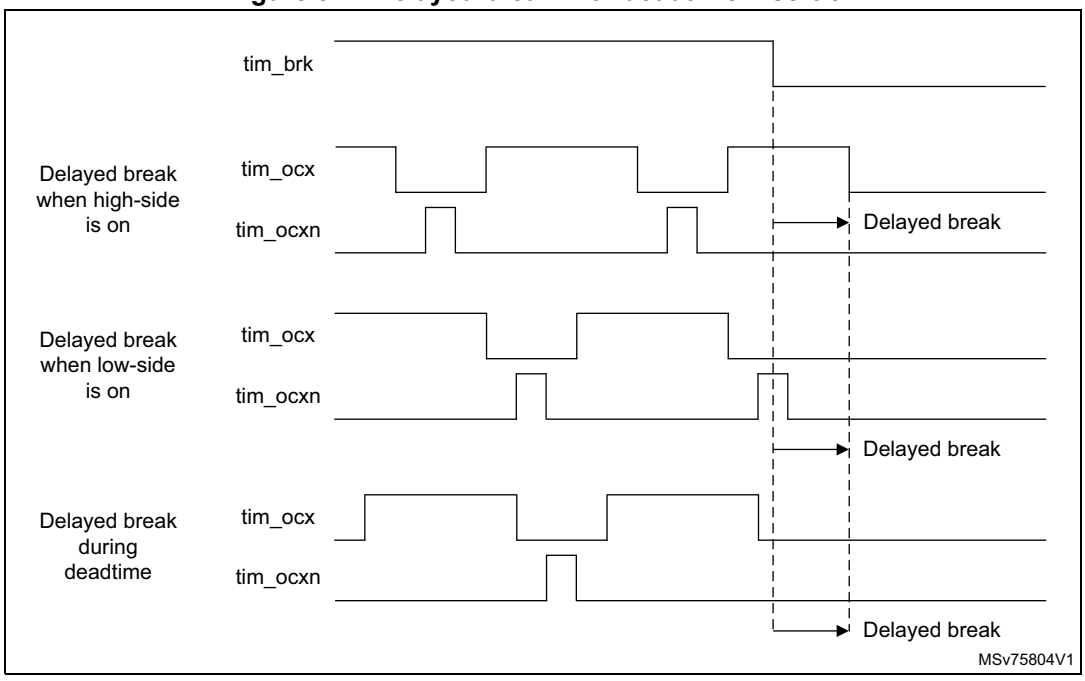
Figure 810. Pulse termination during delayed break



The *Figure 811* depicts the behavior when the deadtime generator is enabled, for the 3 cases:

- Delayed break triggered when tim_ocx is set
- Delayed break triggered when tim_ocxn is set
- Delayed break triggered during deadtime interval

Figure 811. Delayed break with deadtime insertion



The delay programming is identical to deadtime generator's, with 2 user-programmable durations in the TIM_MPR2 register: DBK1[7:0] bitfield for the Delayed 1 break and DBK2[7:0] bitfield for the Delayed 2 break.

DBK1[7:0] and DBK2[7:0] bitfields are preloaded: they can be changed during timer operation, and the new value is transferred to the active register when an update event occurs.

The bits coding for t_{DTG} selection are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBKx[6:0] if DBKx[7] = 0
- DBKx[5:0] if DBKx[7:6] = 10
- DBKx[4:0] if DBKx[7:5] = 110 or 111

The BKMx[1:0] and BKMxN[1:0] bits are static, they can only be modified when the counter is stopped (CEN=0 in TIM_CR1).

The content of the TIM_MPR1 and TIM_MPR2 register can be write protected using the LOCK[1:0] bits in the TIM_BDTR register.

38.4.19 Bidirectional break input

The TIM15/TIM16/TIM17 are featuring bidirectional break I/Os, as represented on *Figure 812*.

They are used to have:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk input is configured in bidirectional mode using the BKBID bit in the TIM_BDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIM_BDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (for example CSS), from on-chip peripherals, or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (triggered by setting the BG bit) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the TIM_BKIN I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 442*).

Table 442. Break protection disarming conditions

MOE	BKBID	BKDSRM	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and rearming break circuitry

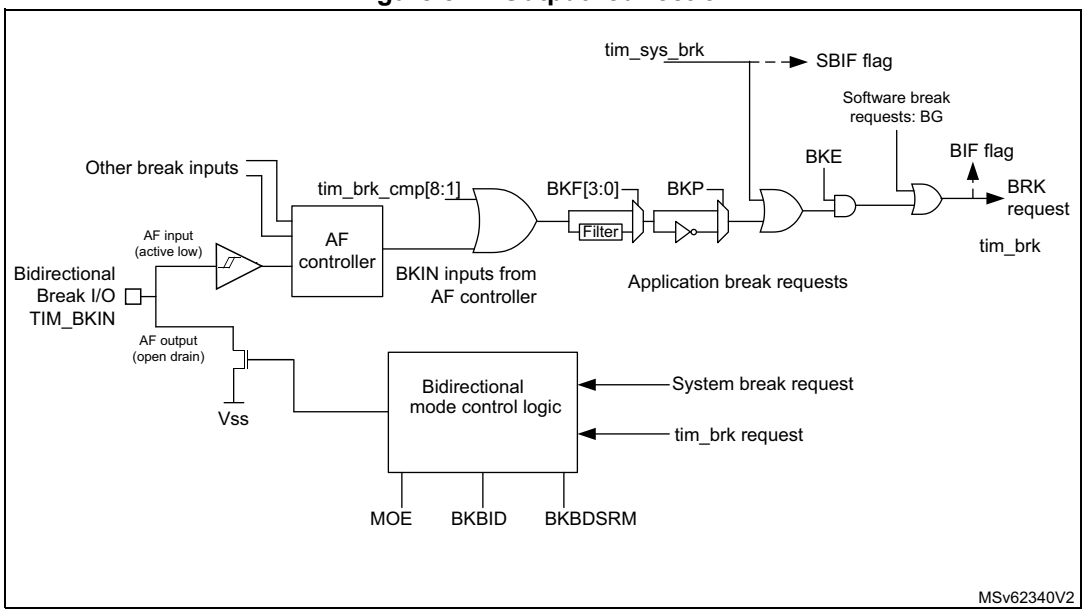
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before rearming).
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 812. Output redirection



38.4.20 Output idle state override

It is possible to override the outputs states when they are disabled (ie MOE bit reset in TIM_BDTR). This feature can be used to apply specific patterns prior to start PWM operations, for instance to refresh bootstrap capacitors of power switch gate drivers.

The OOC control bit in the TIM_OOR register acts as following:

- OOC= 0: 0: tim_ocx and tim_ocxn outputs are in the default idle state
- OOC=1: tim_ocx and tim_ocxn outputs are forced in an idle state defined with OOSx/OOSxN bits in the TIM_OOR register

The OOC bit can only be modified when the outputs are in idle state (MOE = 0). It is cleared by hardware when the MOE bit is set, and the deadtime generator ensures there's no deadtime violation on complementary outputs.

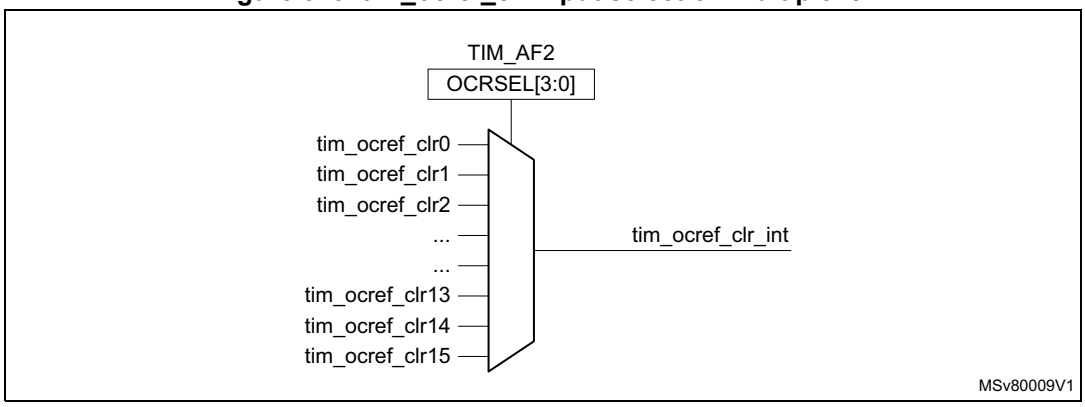
The OOSx/OOSxN bits in the TIM_OOR register cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

38.4.21 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIM_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It must not be used in Forced mode.

The tim_ocref_clr_int input can be selected among several inputs, as shown on *Figure 813* below.

Figure 813. tim_ocref_clr input selection multiplexer



This feature is also supported in combined and asymmetric modes. In this case, it is mandatory to have the OCxCE bit enabled for the channel, the OCxCE bit of the complementary channel (for example OC2CE = X for oc2ref in *Figure 814*) is ignored in this case.

Figure 814 to *Figure 816* are describing the output waveforms for some use cases, when the ocref_clr_int signal is asserted.

Figure 814. Asymmetric PWM mode 1

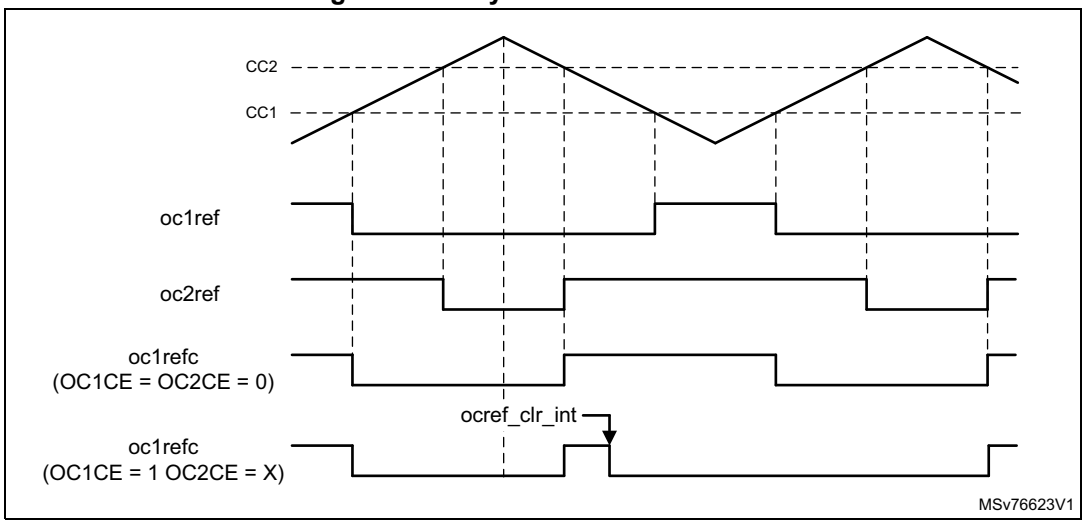


Figure 815. Asymmetric PWM mode 2

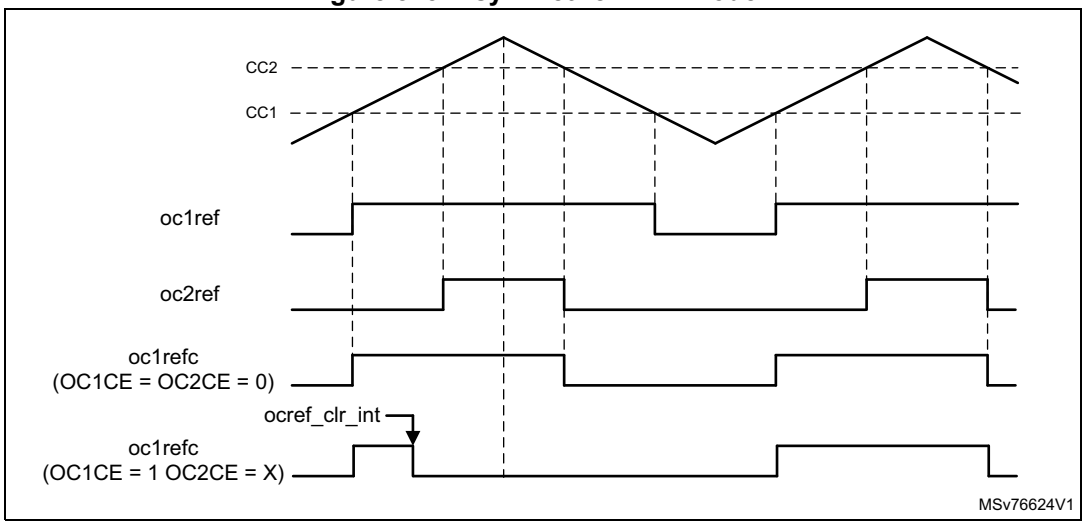
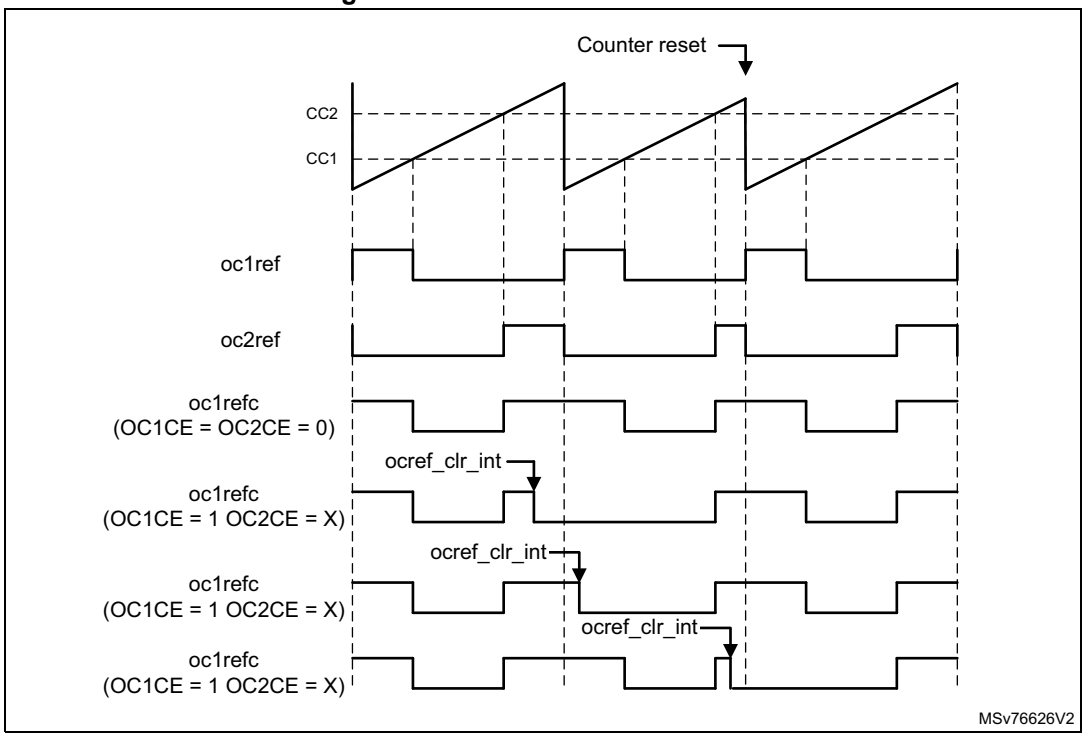


Figure 816. Combined PWM mode 1



Conditions of use

This feature is intended to support cycle-by-cycle current control loops, typically with a comparator shortening the PWM on-time when a current threshold is exceeded.

It cannot be used for the following modes, where the OCxCE bit must be cleared:

- PWM toggle mode (OCxM[4:0] = 0x03)
- One pulse mode with PWM active on match (OPM=1 and OCxM[4:0] = 0x06, 0x08, 0x0C, 0x0E)
- Combined PWM mode 1 in center aligned mode (CMS[1:0] = 1,2,3 and OCxM[4:0] = 0x0C)

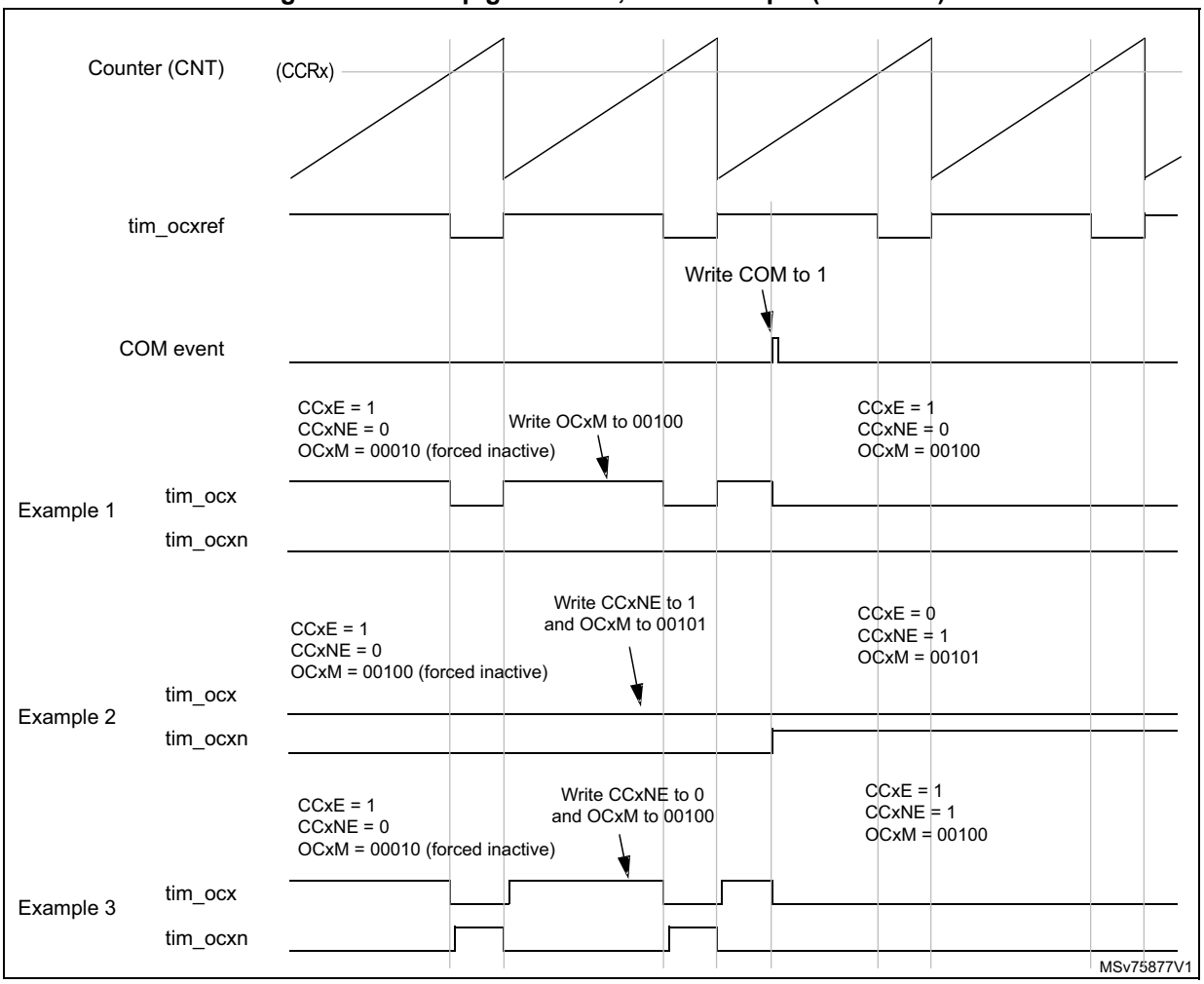
38.4.22 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE, and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIM_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIM_SR register), which can generate an interrupt (if the COMIE bit is set in the TIM_DIER register) or a DMA request (if the COMDE bit is set in the TIM_DIER register).

The *Figure 817* describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 817. 6-step generation, COM example (OSSR = 1)



38.4.23 One-pulse mode

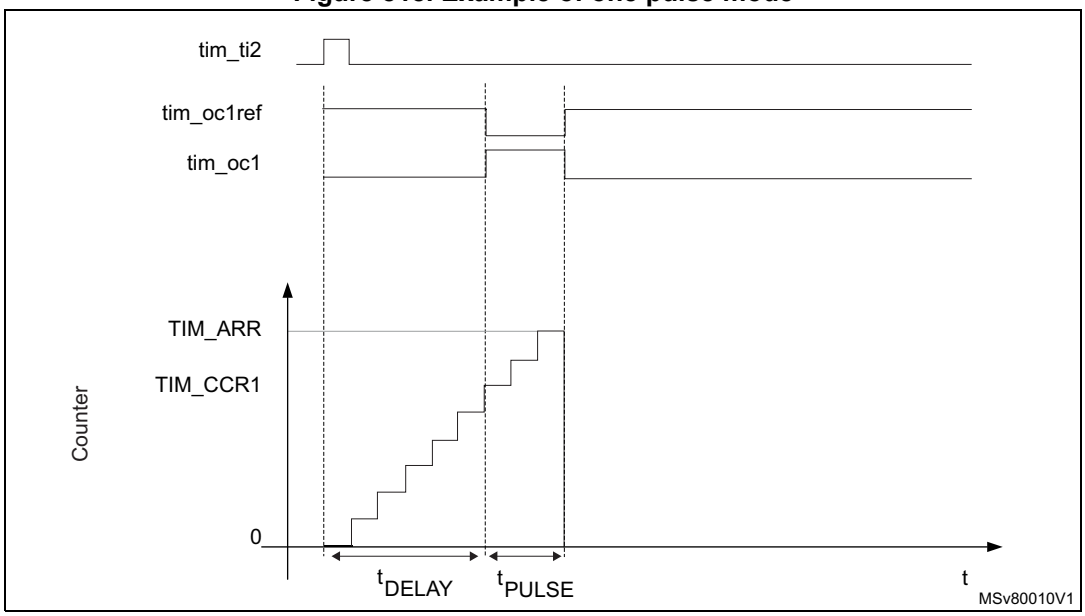
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIM_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 818. Example of one pulse mode



For example one may want to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let's use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:1] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIM_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIM_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 00110 in the TIM_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIM_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIM_ARR - TIM_CCR1).
- Let's say one want to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 00111 in the TIM_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIM_CCMR1 register and ARPE in the TIM_CR1 register. In this case one has to write the compare value in the TIM_CCR1 register, the autoreload value in the TIM_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

Since only one pulse is needed, a 1 must be written in the OPM bit in the TIM_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0).

Particular case: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be obtained.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIM_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

38.4.24 Retriggerable one pulse mode (TIM15 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one pulse mode described in *Section 38.4.23*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

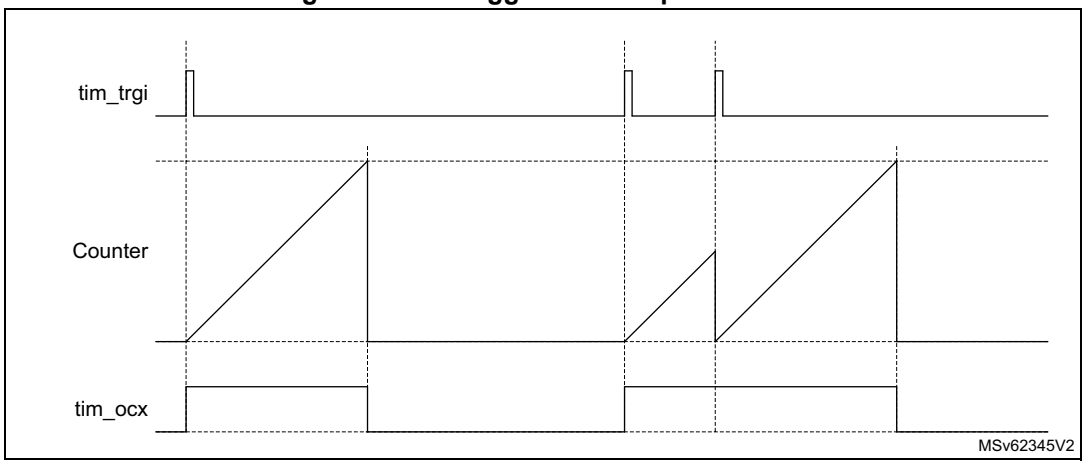
The timer must be in Slave mode, with the bits SMS[4:0] = 01000 (Combined reset + trigger mode) in the TIM_SMCR register, and the OCxM[4:0] bits set to 01000 or 01001 for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[4:0] and SMS[4:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIM_CR1.

Figure 819. Retriggerable one pulse mode



38.4.25 Encoder interface mode

Quadrature encoder

To select Encoder Interface mode write SMS = 00001 in the TIM_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 00010 if it is counting on tim_ti2 edges only and SMS = 00011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIM_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs tim_ti1 and tim_ti2 are used to interface to an quadrature encoder. Refer to *Table 443*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIM_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIM_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIM_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIM_ARR must be configured before starting. In the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 443. Counting direction versus encoder signals (CC1P = CC2P = 0)

Active edge	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	01110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	01111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	00001	High	Down	Up	No count	No count
		Low	Up	Down	No count	No count
Counting on tim_ti2 only x2 mode	00010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	00011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

Figure 820 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example the configuration is the following:

- CC1S = 01 (TIM_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIM_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P = 0 and CC1NP = 0 (TIM_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P = 0 and CC2NP = 0 (TIM_CCER register, tim_ti1fp2 noninverted, tim_ti1fp2= tim_ti2).
- SMS = 00011 (TIM_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIM_CR1 register, Counter enabled).

Figure 820. Example of counter operation in encoder interface mode.

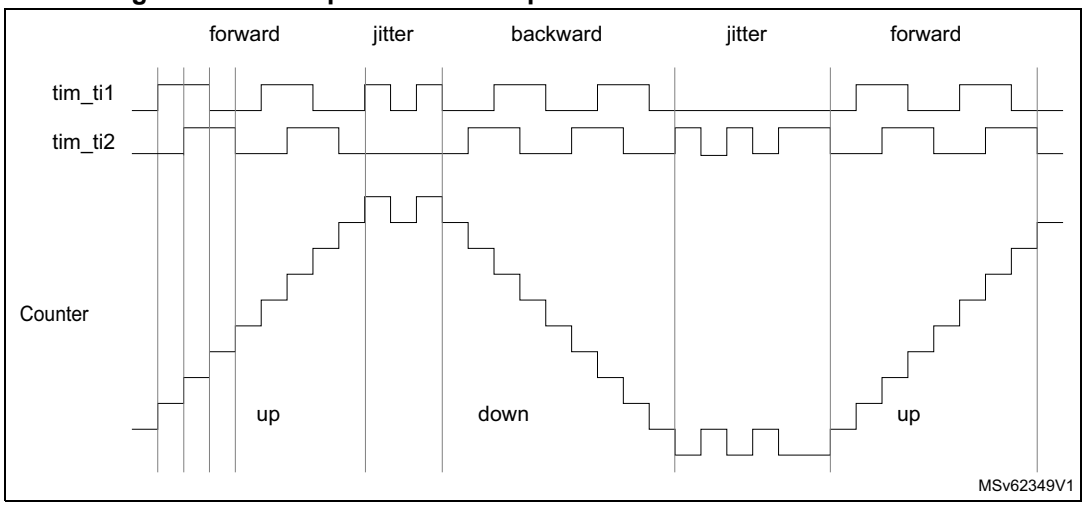


Figure 821 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 821. Example of encoder interface mode with tim_ti1fp1 polarity inverted.

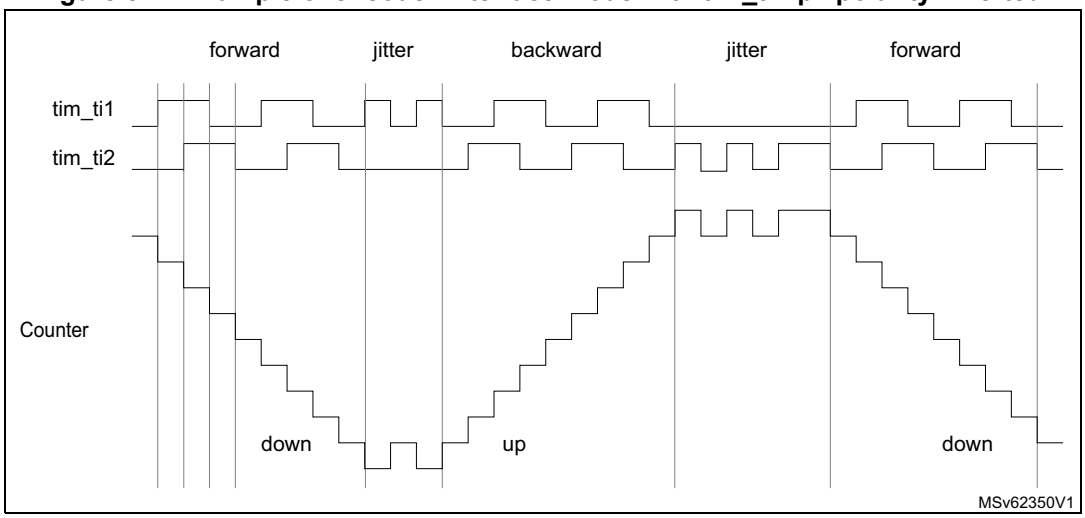
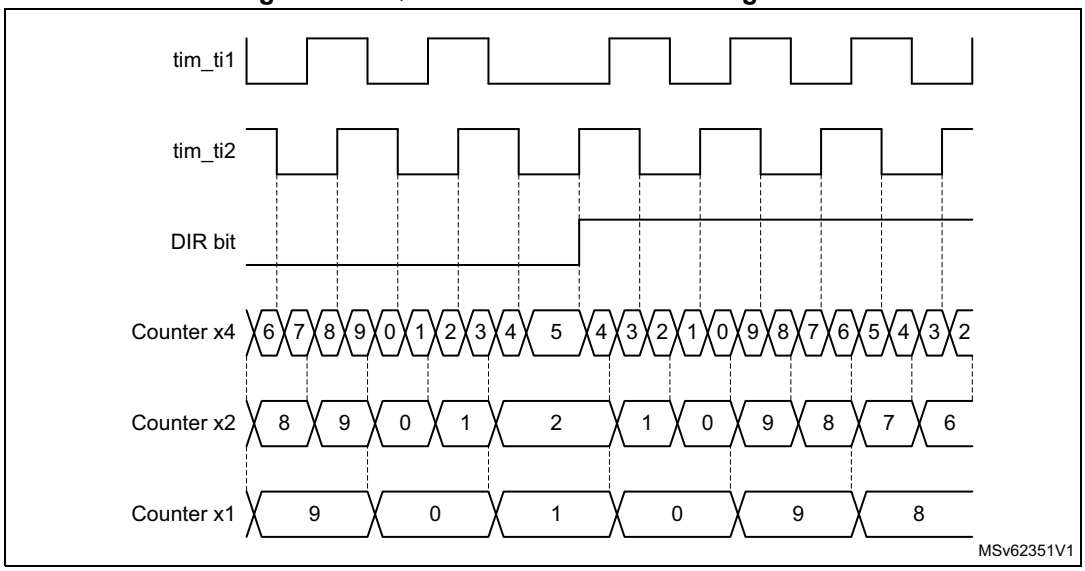


Figure 822 shows the timer counter value during a speed reversal, for various counting modes.

Figure 822. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIM_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

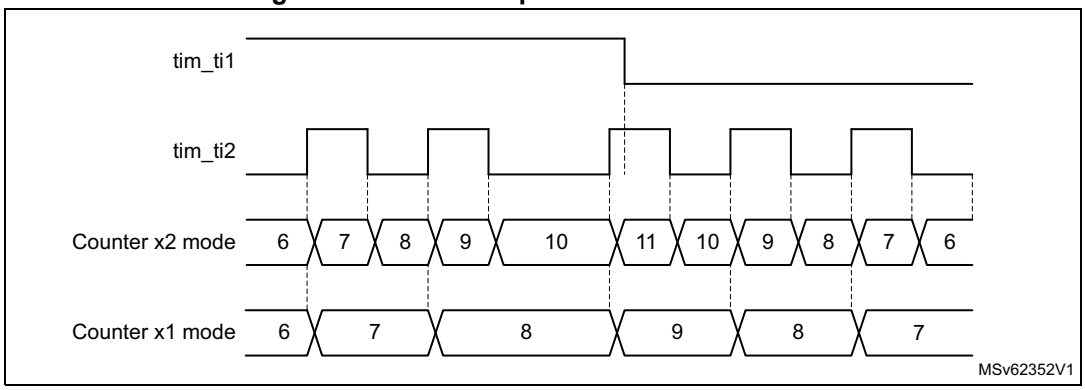
In the clock plus direction mode shown on *Figure 823*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01010: x2 mode, the counter is updated on both rising and falling edges of the clock
- 01011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 823. Direction plus clock encoder mode



Directional clock encoder mode

In the directional clock mode on *Figure 824*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 824*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 825*).
- 01101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 824*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 825*).

Figure 824. Directional clock encoder mode (CC1P = CC2P = 0)

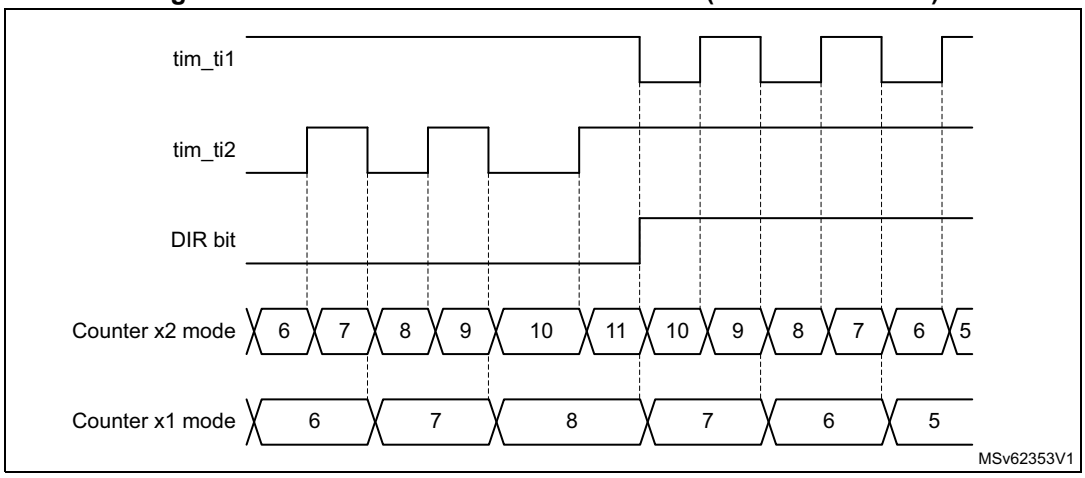


Figure 825. Directional clock encoder mode (CC1P = CC2P = 1)

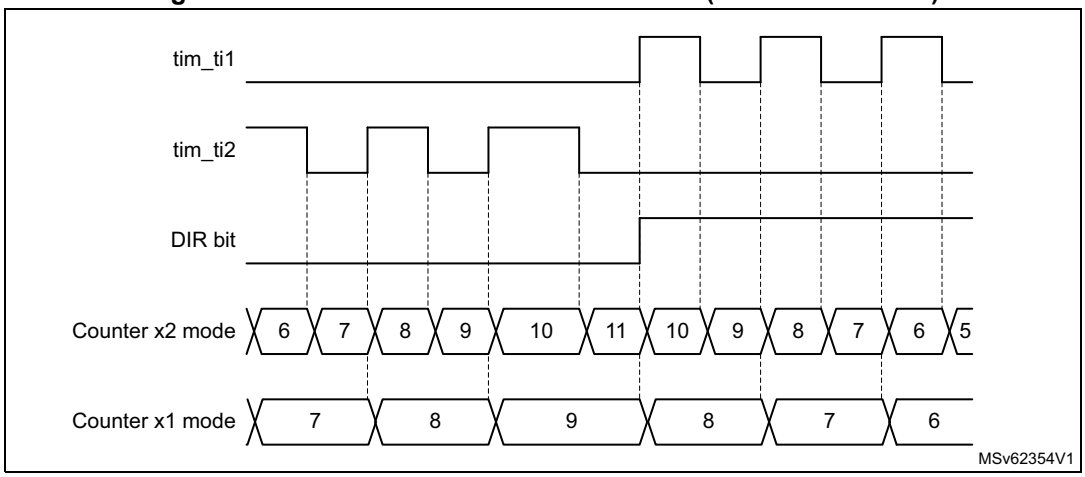


Table 444 here-below details how the directional clock mode operates, for any input transition.

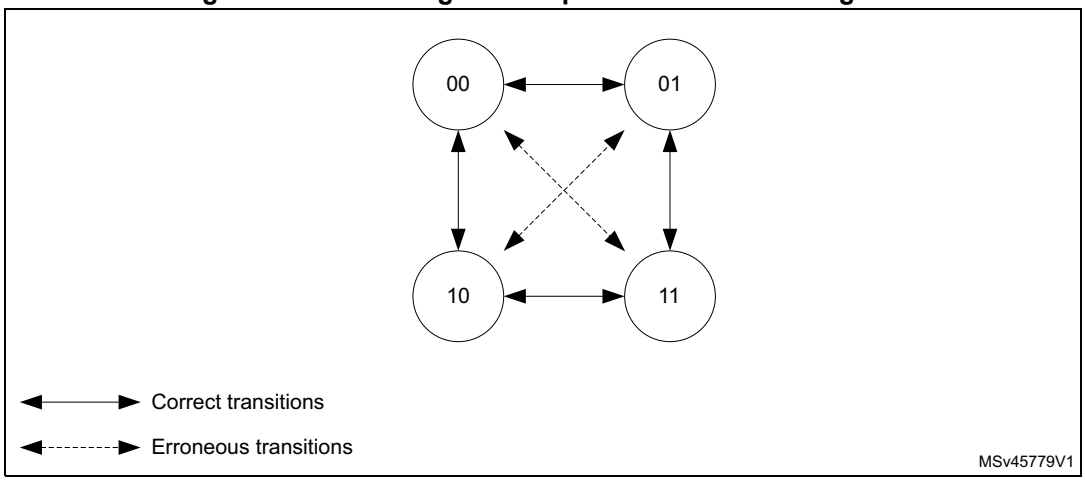
Table 444. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	01100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	01100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	01101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	01101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 826.*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIM_SR status register. A transition error interrupt is generated if the TERRIE bit is set in the TIM_DIER register.

Figure 826. State diagram for quadrature encoded signals



Functional encoder interrupts

The following interrupt is also available in encoder mode

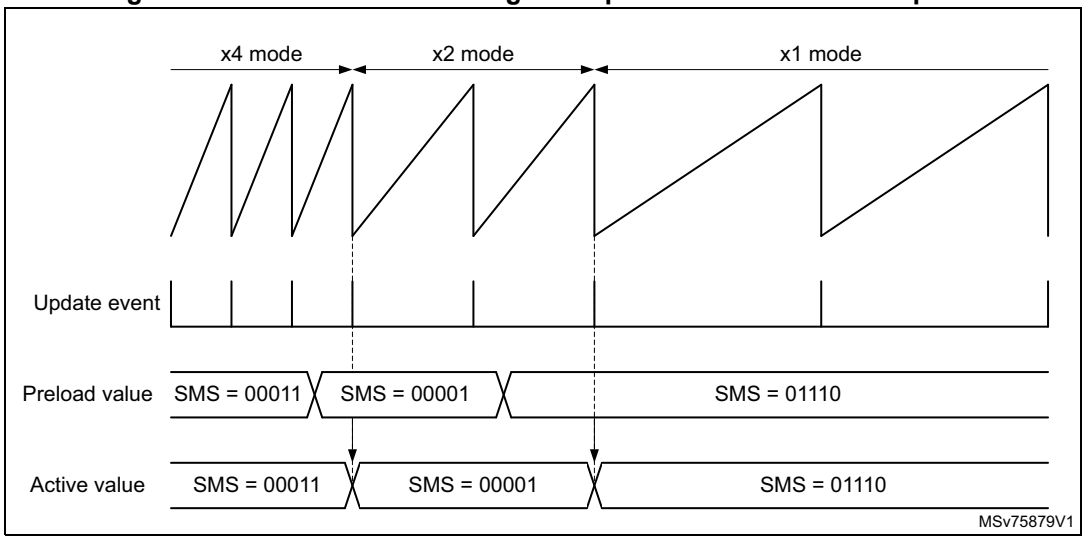
- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIM_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIM_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIM_DIER register.

Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 827*.

For this purpose, the SMS[4:0] bit can be preloaded. The transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting.

Figure 827. Encoder mode change with preload transferred on update



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIM_CR2 register.It is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 10101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

38.4.26 Encoder with built-in debouncer (TIM15 only)

This mode is intended to support noisy digital potentiometers and knobs, providing quadrature encoded signals with potentially long bouncing periods after each transition.

The operating principle consists of working on the first valid transition and discard any other until a second valid transition is detected on the alternate encoder signal.

The *Figure 828* below represents the x2 operating mode. It must be noticed that a transition on input 1 is ignored during direction reversal, if there’s no edge on input 2.

The *Figure 829* below represents the x4 operating mode. In this case, a transition is always ignored during a direction reversal or when the encoder dithers between 2 positions.

These 2 modes are enabled by writing SMS[4:0] = 10000 or 10001 in TIM_SMCR register.

The encoder with built-in debouncer modes are not suitable for tracking a position or an angle in motor control applications. The transitions ignored during direction reversal will lead to a position/angle error that will build-up over time.

Figure 828. Encoder with built-in debouncer, x2 mode

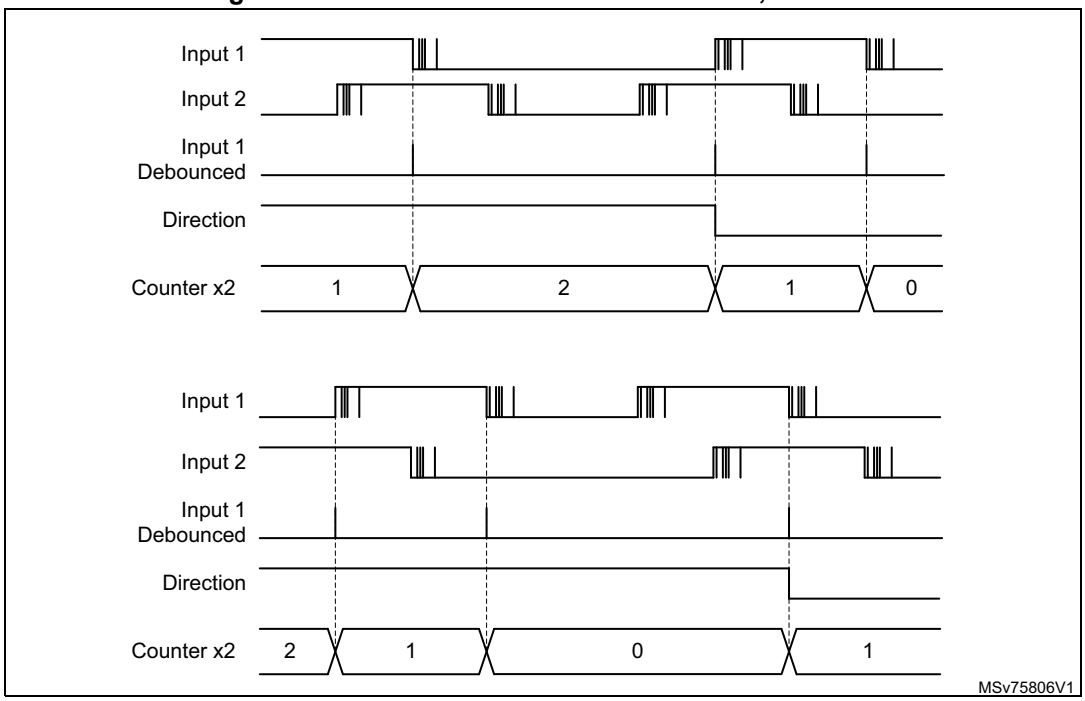
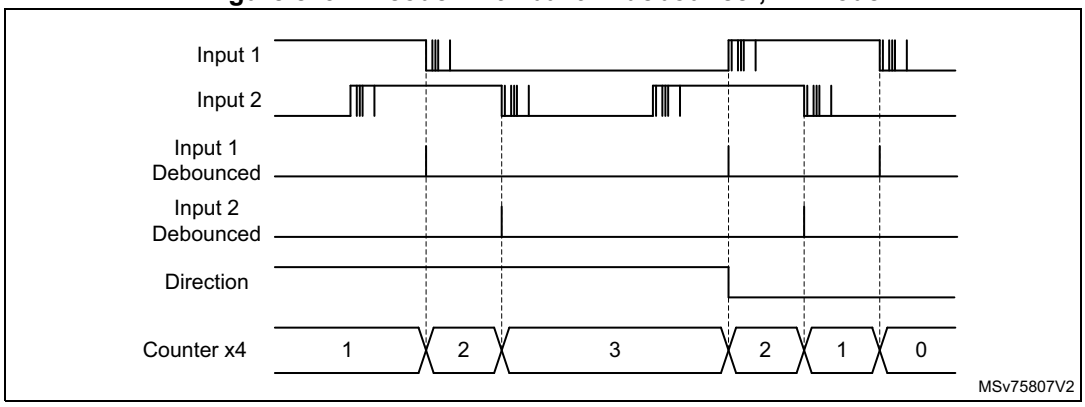


Figure 829. Encoder with built-in debouncer, x4 mode



38.4.27 Clock drift measurement

It is possible to measure the drift between two clock sources using the directional clock encoder mode. This is useful for audio applications typically, when one need to compute the drift between two streams.

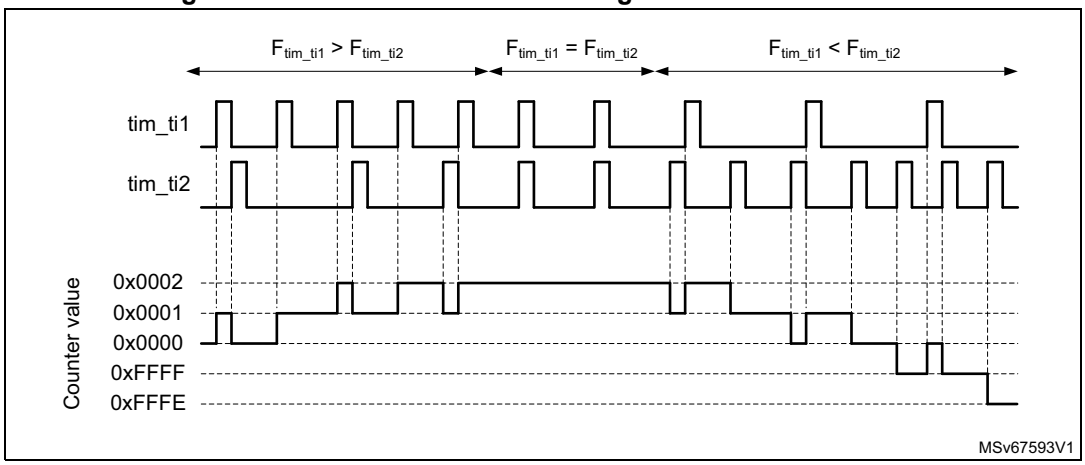
To do so, a signal representing the rate of a first audio stream must be connected to tim_ti1 and a signal representing the rate of a second audio stream must be connected to tim_ti2. The signals representing the audio rates are generally the SAIx_FS_A, SAIx_FS_B, SPIx_WS or spdifrx_frame_sync (this list of signals is provided as example and depends on the audio peripheral availability for a given product line, as detailed on the product datasheet).

In this case, the clocks are present on both tim_ti1 and tim2 simultaneously, as shown on *Figure 830*, and the counter value indicates the drift.

Let's define F_{tim_ti1} and F_{tim_ti2} the frequencies of the signals on tim_ti1 and tim_ti2. The counter value changes as following:

- If $F_{tim_ti1} > F_{tim_ti2}$ the counter value increases
- If $F_{tim_ti1} < F_{tim_ti2}$ the counter value decreases
- If $F_{tim_ti1} = F_{tim_ti2}$ the counter value dithers between two consecutive values, from one clock edge to the other. If the two clocks are perfectly synchronized (clock edges occuring simultaneously on the two inputs), the counter value does not change (as shown on *Figure 830* below).

Figure 830. Clock drift measure using directional clock mode



38.4.28 UIF bit remapping

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag UIF into bit 31 of the timer counter register (TIM_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

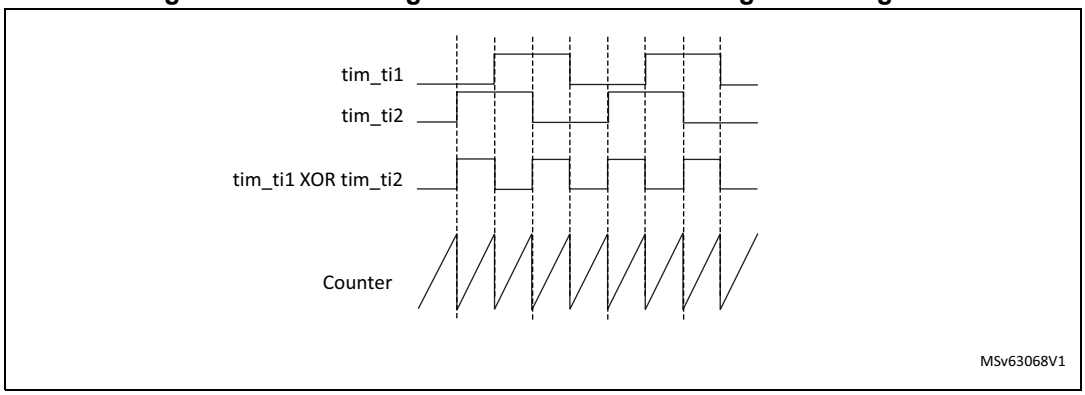
There is no latency between the assertions of the UIF and UIFCPY flags.

38.4.29 Timer input XOR function (TIM15 only)

The TI1S bit in the TIM_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 831*.

Figure 831. Measuring time interval between edges on 2 signals



The XORPS control bit selects the position of the XOR gate, as shown in the *Figure 832* and *Figure 833* below, when the TI1S bit is set.

Figure 832. XOR gate connection with TI1S = 1 and XORPS = 0

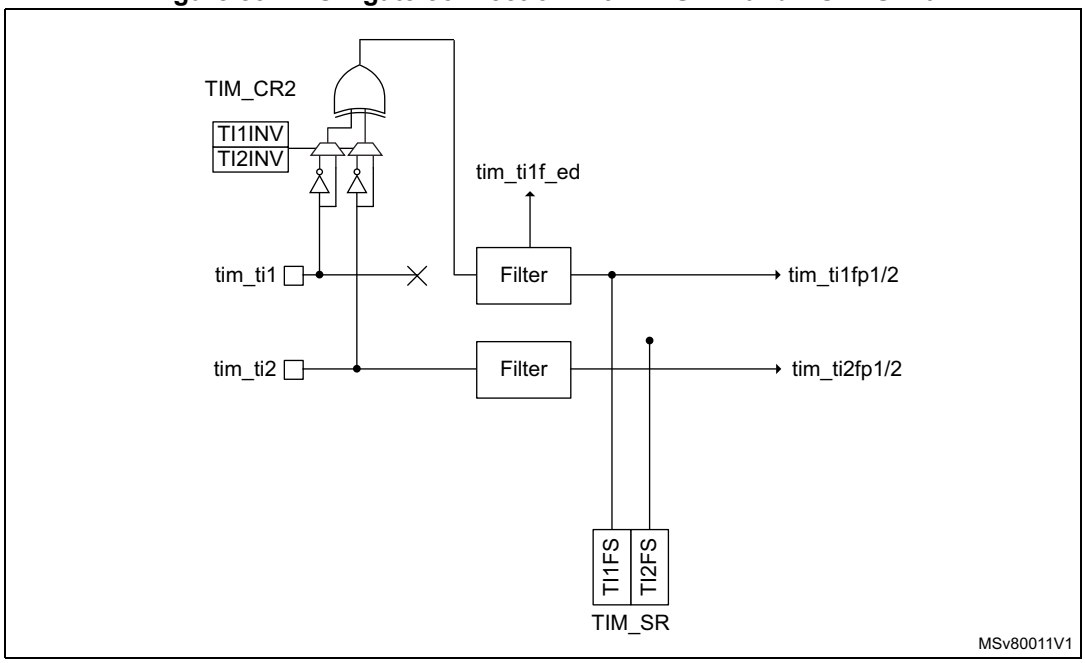
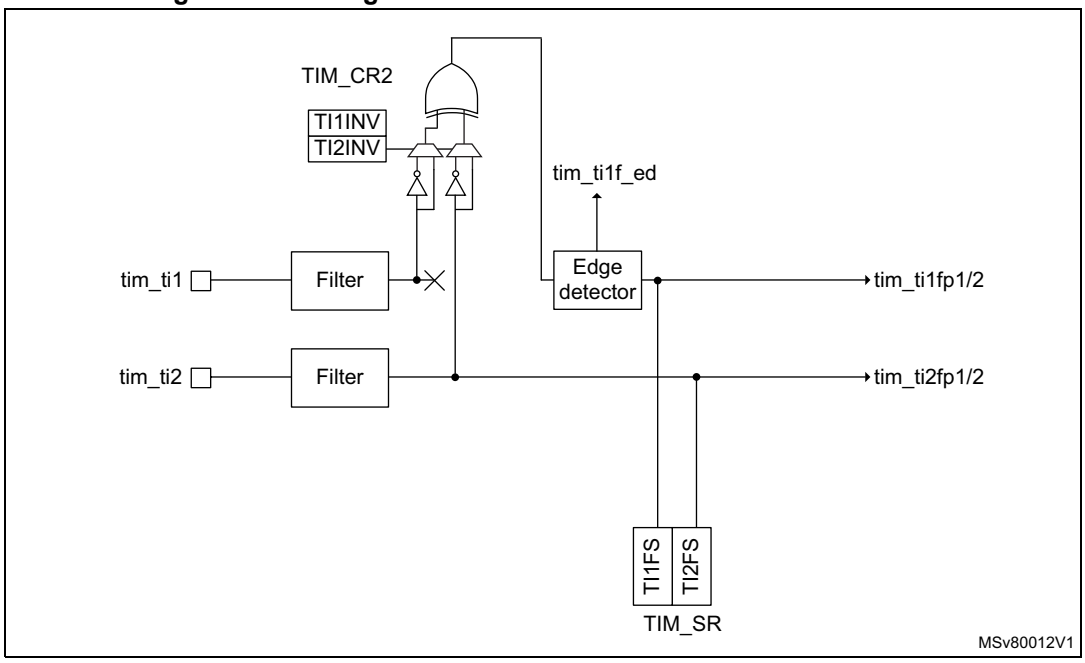


Figure 833. XOR gate connection with TI1S = 1 and XORPS = 1



Two control bits TI1INV and TI2INV in the TIM_CR2 register can invert the tim_in1 and tim_in2 signals prior entering the XOR gate.

The status of the tim_in1 and tim_in2 signals can be software polled using the TI1FS and TI2FS bits in the TIM_SR register.

38.4.30 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIM_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIM_ARR, TIM_CCRx) are updated.

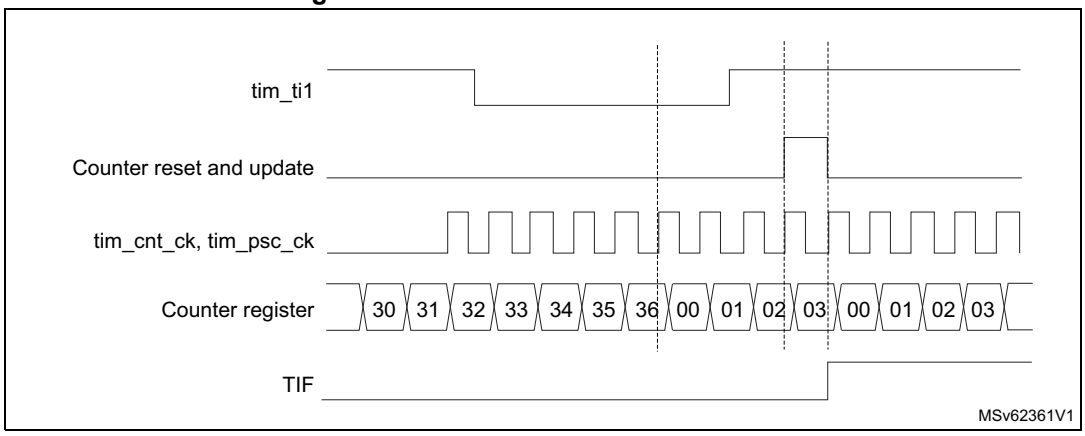
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, no need for any filter, so IC1F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIM_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in the TIM_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 00100 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Start the counter by writing CEN = 1 in the TIM_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIM_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIM_DIER register).

The following figure shows this behavior when the autoreload register TIM_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 834. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

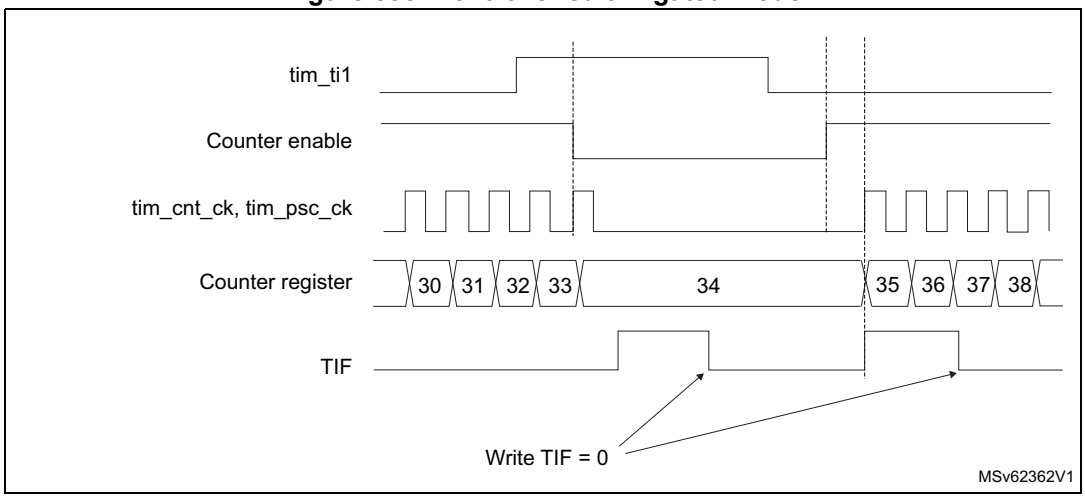
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, no need for any filter, so IC1F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIM_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in the TIM_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 00101 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIM_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIM_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 835. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

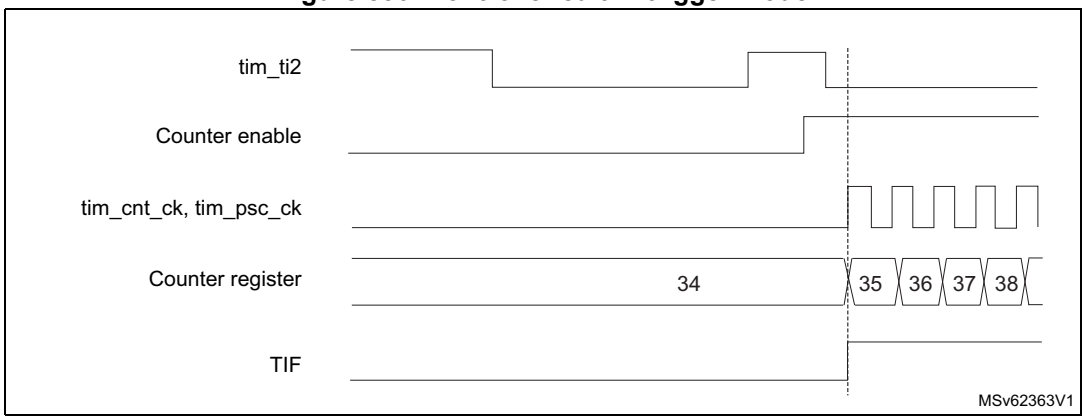
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, no need for any filter, so IC2F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIM_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in the TIM_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 00110 in the TIM_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in the TIM_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 836. Control circuit in trigger mode



Slave mode selection preload for run-time encoder mode update

The SMS[4:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIM_SMCR register. The trigger for the transfer from SMS[4:0] preload to active value is the update event (UEV) occurring when the counter overflows.

38.4.31 Slave mode – combined reset + trigger mode (TIM15 only)

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

38.4.32 Slave mode – combined reset + gated mode (TIM15 only)

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

38.4.33 Timer synchronization (TIM15 only)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 35.4.27: Timer synchronization* for details.

Note: The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

38.4.34 Using timer output as trigger for other timers (TIM16/TIM17 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

38.4.35 ADC triggers (TIM15 only)

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

38.4.36 ADC synchronization (TIM15 only)

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIM_CR2 register.

This feature is useful when the timers and the ADCs are operating with semi-synchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- when the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum
- when the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the 2 requirements below is met (depending on the selected trigger source):

1. The counter period must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times (\text{TIM_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times \text{TIM_CCRx} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

If none of the 2 above requirement are met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock
2. Configure the timer and set the ADSYNC bit
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits)
4. Start the timer (with the CEN counter enable bit)

38.4.37 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to reprogram several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIM_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIM_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIM_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIM_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register.

Example:

00000: TIM_CR1

00001: TIM_CR2

00010: TIM_SMCR

The DBSS[3:0] bits in the TIM_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 38.7.23: TIM output override register (TIM_OOR)* and *Section 38.8.21: TIM output override register (TIM_OOR)* for details).

For example, the timer DMA burst feature can be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 2, DBA = 0xE and DBSS[3:0] = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

38.4.38 TIM15/TIM16/TIM17 DMA requests

The TIM15/TIM16/TIM17 can generate a DMA request, as shown in *Table 445*.

Table 445. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_com_dma	Commutation (COM)	COMDE
tim_trgi_dma	Trigger	TDE

38.4.39 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

For more details, refer to the debug section.

38.5 TIM15/TIM16/TIM17 low-power modes

Table 446. Effect of low-power modes on TIM15/TIM16/TIM17

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

38.6 TIM15/TIM16/TIM17 interrupts

The TIM15/TIM16/TIM17 can generate multiple interrupts, as shown in *Table 447*.

Table 447. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UP	Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2(1)	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
TIM_TRG_COM	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	Trigger(1)	TIF	TIE	write 0 in TIF	Yes	No
TIM_DIR	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_BRK	Break	BIF	BIE	write 0 in BIF	Yes	No
	System Break	SBIF		write 0 in SBIF	Yes	No
TIM_TER	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

1. Available for TIM15 only.

38.7 TIM15 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

38.7.1 TIM control register 1 (TIM_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CKD2[3:0]**: Clock division 2

This bit field indicates the division between sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_etr_in, tim_tix).

0000: No Division
0001: $t_{DTS2} = 4*t_{DTS}$
0010: $t_{DTS2} = 16*t_{DTS}$
0011: $t_{DTS2} = 64*t_{DTS}$
0100: $t_{DTS2} = 256*t_{DTS}$
0101: $t_{DTS2} = 1024*t_{DTS}$
0110: $t_{DTS2} = 4096*t_{DTS}$
0111: $t_{DTS2} = 16384*t_{DTS}$
1000: $t_{DTS2} = 65536*t_{DTS}$
1001: $t_{DTS2} = 262144*t_{DTS}$
Others: Reserved, do not program this value

Bits 23:13 Reserved, must be kept at reset value.

Bit 12 DITHEN: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_tix)

00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: $t_{DTS} = 8*t_{tim_ker_ck}$

Bit 7 **ARPE**: Auto-reload preload enable

0: TIM_ARR register is not buffered
1: TIM_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1).

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt if enabled. These events can be:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

38.7.2 TIM control register 2 (TIM_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TI2INV	TI1INV	ADSYNC	XORPS	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw		rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TI2INV**: tim_ti2 signal inversion on XOR gate input

0: tim_ti2 non-inverted

1: tim_ti2 inverted

Bit 29 **TI1INV**: tim_ti1 signal inversion on XOR gate input

0: tim_ti1 non-inverted

1: tim_ti1 inverted

Bit 28 **ADSYNC**: ADC synchronization
0: The timer operates independently from the ADC
1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 38.4.36* for requirements.
The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bit 27 **XORPS**: XOR gate position selection
This bit defines the position of the XOR gate with respect to input filters.
0: XOR gate placed before TI1 filter
1: XOR gate placed after TI1 and TI2 filters

Bit 26 Reserved, must be kept at reset value.

Bits 24:11 Reserved, must be kept at reset value.

Bit 10 **OIS2:** Output idle state 2 (tim_oc2 output)
0: tim_oc2 = 0 when MOE = 0
1: tim_oc2 = 1 when MOE = 0
Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIM_BKR register).

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)
0: tim_oc1n = 0 after a dead-time when MOE = 0
1: tim_oc1n = 1 after a dead-time when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)
0: tim_oc1 = 0 after a dead-time when MOE = 0
1: tim_oc1 = 1 after a dead-time when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BKR register).

Bit 7 **TI1S**: tim_ti1 selection
0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input
1: The tim_ti1_in[15:0] and tim_ti2_in[15:0] multiplexers output are connected to the tim_ti1 input (XOR combination)

Bits 25, 6:4 **MMS[3:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIM_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo).

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo).

1000: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

Others: reserved.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

38.7.3 TIM slave mode control register (TIM_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **SMSPE:** SMS preload enable

This bit selects whether the SMS[4:0] bitfield is preloaded.

0: SMS[4:0] bitfield is not preloaded

1: SMS[4:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:18 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/slave mode

0: No action

1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if the user wants to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: Reserved
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
10100: Internal trigger 16 (tim_itr16)
10101: Internal trigger 17 (tim_itr17)
10110: Internal trigger 18 (tim_itr18)
10111: Internal trigger 19 (tim_itr19)
11000: Internal trigger 20 (tim_itr20)
11001: Internal trigger 21 (tim_itr21)
11010: Internal trigger 22 (tim_itr22)
11011: Internal trigger 23 (tim_itr23)
11100: Internal trigger 24 (tim_itr24)
11101: Internal trigger 25 (tim_itr25)
11110: Internal trigger 26 (tim_itr26)
11111: Internal trigger 27 (tim_itr27)
Others: Reserved

See *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for more details on tim_itrx meaning for each timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 17:16, 2:0 **SMS[4:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIM_SMCR for tim_etr_in and CCxP/CCxNP bits in TIM_CCER register for tim_ti1fp1 and tim_ti2fp2).

00000:Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

00001: Quadrature encoder mode 1, x2 mode- Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

00010:Quadrature encoder mode 2, x2 mode - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

00011:Quadrature encoder mode 3, x4 mode - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

00100:Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

00101:Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

00110:Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

00111:External Clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

01000:Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

01001:Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

01010:Encoder mode: Clock plus direction, x2 mode.

01011:Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P

01100:Encoder mode: Directional Clock, x2 mode.

01101:Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

01110:Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

01111:Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

10000:Quadrature encoder with built-in debouncer: x2 mode.

10001:Quadrature encoder with built-in debouncer: x4 mode.

Others: Reserved.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

38.7.4 TIM DMA/interrupt enable register (TIM_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	Res.	DIRIE	Res.	Res.	Res.	Res.	Res.
								rw		rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	rw	rw			rw	rw	rw	rw	rw	rw			rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 Reserved, must be kept at reset value.

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction Change interrupt disabled
1: Direction Change interrupt enabled

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

38.7.5 TIM status register (TIM_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	TERRF	Res.	DIRF	Res.	Res.	Res.	Res.	Res.
		r	r				rc_w0	rc_w0		rc_w0					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	BGF	SBIF	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
	rc_w0	rc_w0			rc_w0	rc_w0		rc_w0	rc_w0	rc_w0			rc_w0	rc_w0	rc_w0

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **TI2FS**: tim_ti2f status
This flag indicates the value of the tim_ti2f signal (after the digital filtering stage), for polling purpose.
0: tim_ti2f signal is low
1: tim_ti2f signal is high

Bit 28 **TI1FS**: tim_ti1f status
This flag indicates the value of the tim_ti1f signal (after the digital filtering stage), for polling purpose.
0: tim_ti1f signal is low
1: tim_ti1f signal is high

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **UIOVRF**: Update Interrupt Overrun flag
This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.
0: No Update Interrupt Overrun
1: Update Interrupt Overrun

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 Reserved, must be kept at reset value.

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIM_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 **BGF**: Break generation flag
This flag is set by hardware as soon as a break event is forced by software, using the BG bit. It is useful to distinguish hardware-triggered from software-triggered break events. It is cleared by software by writing it to 0.
0: No break event occurred.
1: A break event was forced with the BG bit

Bit 13 **SBIF**: System break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE = 1 in the TIM_DIER register.

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected
1: The counter value has been captured in TIM_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag
This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred
1: Trigger interrupt pending

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/compare 2 interrupt flag

refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIM_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred

If channel CC1 is configured as output: this flag is set when the content of the counter TIM_CNT matches the content of the TIM_CCR1 register. When the content of TIM_CCR1 is greater than the content of TIM_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIM_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIM_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIM_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIM_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.

– When CNT is reinitialized by a trigger event (refer to *Section 38.7.3: TIM slave mode control register (TIM_SMCR)*), if URS = 0 and UDIS = 0 in the TIM_CR1 register.

38.7.6 TIM event generation register (TIM_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
								w	w	rw			w	w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIM_SR register. Related interrupt or DMA transfer can occur if enabled

Bit 5 **COMG:** Capture/compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1 A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIM_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

38.7.7 TIM capture/compare mode register 1 (TIM_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIM_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

38.7.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC2M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]	
						rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

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

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable
Refer to OC1CE description on bit 7.

Bits 25:24, 14:12 **OC2M[4:0]**: Output compare 2 mode
Refer to OC1M description on bits 6:4.

Bit 11 **OC2PE**: Output compare 2 preload enable
Refer to OC1PE description on bit 3.

Bit 10 **OC2FE**: Output compare 2 fast enable
Refer to OC1FE description on bit 2.

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output.
01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.
10: C2 channel is configured as input, tim_ic2 is mapped on tim_ti1.
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable
0: tim_oc1ref is not affected by the tim_ocref_clr_int input.
1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input.
Note: This mode is not available for all PWM modes. Refer to Section 38.4.21: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC1M[4:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

00000: Frozen - The comparison between the output compare register TIM_CCR1 and the counter TIM_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

00001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00011: Toggle - tim_oc1ref toggles when TIM_CNT = TIM_CCR1.

00100: Force inactive level - tim_oc1ref is forced low.

00101: Force active level - tim_oc1ref is forced high.

00110: PWM mode 1 - Channel 1 is active as long as TIM_CNT<TIM_CCR1 else inactive.

00111: PWM mode 2 - Channel 1 is inactive as long as TIM_CNT<TIM_CCR1 else active.

01000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01010: Reserved

01011: Reserved

01100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

01101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

01110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

01111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

On channels that have a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIM_CCR1 disabled. TIM_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR1 enabled. Read/Write operations access the preload register. TIM_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

38.7.9 TIM capture/compare enable register (TIM_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								rw		rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/compare 1 output polarity

CC1 channel configured as output:

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 448* for details.

Table 448. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15)

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	tim_ocx output state	tim_ocxn output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) tim_ocx = 0 tim_ocxn = 0	
		1	0	0	Off-State (output enabled with inactive state) tim_ocx=CCxP, tim_ocxn=CCxNP	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) tim_ocx = 0	tim_ocxref + Polarity tim_ocxn = tim_ocxref XOR CCxNP
		0	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) tim_ocxn = 0
		X	1	1	tim_ocxref + Polarity + dead-time	Complementary to tim_ocxref (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_ocx = CCxP	tim_ocxref + Polarity tim_ocxn = tim_ocxref XOR CCxNP
		1	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref xor CCxP	Off-State (output enabled with inactive state) tim_ocxn = CCxNP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z)	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_ocx = CCxP, tim_ocxn = CCxNP Then if the clock is present: tim_ocx = OISx and tim_ocxn = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to tim_ocx and tim_ocxn both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and GPIO control and alternate function selection registers.

38.7.10 TIM counter (TIM_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIM_SR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

38.7.11 TIM prescaler (TIM_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency ($f_{tim_cnt_ck}$) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM_EGR register or through trigger controller when configured in “reset mode”).

38.7.12 TIM autoreload register (TIM_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 38.4.3: Time-base unit on page 2231* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

38.7.13 TIM repetition counter register (TIM_RCR)

Address offset: 0x30

Reset value: 0x0000

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

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the reptition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

– The number of PWM periods in edge-aligned mode.
– The number of half PWM period in center-aligned mode.

38.7.14 TIM capture/compare register 1 (TIM_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIM_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

38.7.15 TIM capture/compare register 2 (TIM_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signalled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIM_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

38.7.16 TIM break and dead-time register (TIM_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIM_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample the tim_brk input signal and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_ocx and tim_ocxn outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_ocx and tim_ocxn outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIM_CCER register)

See tim_ocx/tim_ocxn enable description for more details (*Section 38.7.9: TIM capture/compare enable register (TIM_CCER) on page 2319*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

Note: This bit can not be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk clock failure event) disabled

1: Break inputs (tim_brk and tim_sys_brk clock failure event) enabled

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_ocx/tim_ocxn enable description for more details (*Section 38.7.9: TIM capture/compare enable register (TIM_CCER) on page 2319*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_ocx/tim_ocxn outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE = 0 on channels configured as outputs.

See tim_ocx/tim_ocxn enable description for more details (*Section 38.7.9: TIM capture/compare enable register (TIM_CCER) on page 2319*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (tim_ocx/tim_ocxn enable output signal = 0)

1: When inactive, tim_ocx/tim_ocxn outputs are forced first with their idle level as soon as CCxE = 1 or CCxNE = 1. tim_ocx/tim_ocxn enable output signal = 1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIM_BDTR register, OISx and OISxN bits in TIM_CR2 register and BKBID/BKE/BKP/AOE bits in TIM_BDTR register can no longer be written

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIM_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: OCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIM_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIM_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx → DT = DTG[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$

DTG[7:5] = 10x → DT = (64+DTG[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$

DTG[7:5] = 110 → DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$

DTG[7:5] = 111 → DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$

Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

38.7.17 TIM timer deadtime register 2 (TIM_DTR2)

Address offset: 0x054

Reset value: 0x0000 0000

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

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

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

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx → DTF = DTGF[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$.

DTGF[7:5] = 10x → DTF = (64+DTGF[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$.

DTGF[7:5] = 110 → DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$.

DTGF[7:5] = 111 → DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$.

Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

38.7.18 TIM input selection register (TIM_TISEL)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects tim_ti2_in[15:0] input

0000: TIM15_CH2 input (tim_ti2_in0)

0001: tim_ti2_in1

...

1111: tim_ti2_in15

Refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input

0000: TIM15_CH1 input (tim_ti1_in0)

0001: tim_ti1_in1

...

1111: tim_ti1_in15

Refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

38.7.19 TIM alternate function register 1 (TIM_AF1)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKCMP20E	BKCMP19E	BKCMP18E	BKCMP17E	BKCMP16E	BKCMP15E	BKCMP14E	BKCMP13E	BKCMP12E	BKCMP11E	BKCMP10E	BKCMP9E	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Bit 31 **BKCMP20E**: tim_brk_cmp20 enable

This bit enables the tim_brk_cmp20 for the timer's tim_brk input. tim_brk_cmp20 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp20 input disabled

1: tim_brk_cmp20 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 30 **BKCMP19E**: tim_brk_cmp19 enable

This bit enables the tim_brk_cmp19 for the timer's tim_brk input. tim_brk_cmp19 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp19 input disabled

1: tim_brk_cmp19 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 29 **BKCMP18E**: tim_brk_cmp18 enable

This bit enables the tim_brk_cmp18 for the timer's tim_brk input. tim_brk_cmp18 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp18 input disabled

1: tim_brk_cmp18 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 28 **BKCMP17E**: tim_brk_cmp17 enable

This bit enables the tim_brk_cmp17 for the timer's tim_brk input. tim_brk_cmp17 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp17 input disabled

1: tim_brk_cmp17 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 27 **BKCMP16E**: tim_brk_cmp16 enable

This bit enables the tim_brk_cmp16 for the timer's tim_brk input. tim_brk_cmp16 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp16 input disabled

1: tim_brk_cmp16 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 26 **BKCMP15E**: tim_brk_cmp15 enable

This bit enables the tim_brk_cmp15 for the timer's tim_brk input. tim_brk_cmp15 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp15 input disabled

1: tim_brk_cmp15 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 25 **BKCMP14E**: tim_brk_cmp14 enable

This bit enables the tim_brk_cmp14 for the timer's tim_brk input. tim_brk_cmp14 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp14 input disabled

1: tim_brk_cmp14 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 24 **BKCMP13E**: tim_brk_cmp13 enable

This bit enables the tim_brk_cmp13 for the timer's tim_brk input. tim_brk_cmp13 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp13 input disabled

1: tim_brk_cmp13 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 23 **BKCMP12E**: tim_brk_cmp12 enable

This bit enables the tim_brk_cmp12 for the timer's tim_brk input. tim_brk_cmp12 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp12 input disabled

1: tim_brk_cmp12 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 22 **BKCMP11E**: tim_brk_cmp11 enable

This bit enables the tim_brk_cmp11 for the timer's tim_brk input. tim_brk_cmp11 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp11 input disabled

1: tim_brk_cmp11 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 21 **BKCMP10E**: tim_brk_cmp10 enable

This bit enables the tim_brk_cmp10 for the timer's tim_brk input. tim_brk_cmp10 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp10 input disabled

1: tim_brk_cmp10 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 20 **BKCMP9E**: tim_brk_cmp9 enable

This bit enables the tim_brk_cmp9 for the timer's tim_brk input. tim_brk_cmp9 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp9 input disabled

1: tim_brk_cmp9 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 19:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input is active high

1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

38.7.20 TIM alternate function register 2 (TIM_AF2)

Address offset: 0x064

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **OCRSEL[3:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

0000: tim_ocref_clr0

0001: tim_ocref_clr1

...

1111: tim_ocref_clr15

Refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

38.7.21 TIM multilevel protection register 1 (TIM_MPR1)

Address offset: 0x090

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2M[1:0]		BK1NM[1:0]		BK1M[1:0]	
										rw	rw	rw	rw	rw	rw

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

Bits 5:4 **BK2M[1:0]**: Break mode for tim_oc2 output

This bit must be modified only when CEN=0 in TIM_CR1.

00: Immediate break

01: Delayed 1 break

10: Delayed 2 break

11: Reserved

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 3:2 **BK1NM[1:0]**: Break mode for tim_oc1n output

This bit must be modified only when CEN=0 in TIM_CR1.

00: Immediate break

01: Delayed 1 break

10: Delayed 2 break

11: Reserved

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 1:0 **BK1M[1:0]**: Break mode for tim_oc1 output

This bit must be modified only when CEN=0 in TIM_CR1.

00: Immediate break

01: Delayed 1 break

10: Delayed 2 break

11: Reserved

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

38.7.22 TIM multilevel protection register 2 (TIM_MPR2)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **DBK2[7:0]**: Delayed 2 Break duration

This bitfield defines the duration of the delay in Delayed 2 Break mode.

It is partially preloaded and can be updated during timer operation (new value taken into account after an update event), at the exception of the bits coding for Tdtg selection, which are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBK2[6:0] if DBK2[7] = 0
- DBK2[5:0] if DBK2[7:6] = 10
- DBK2[4:0] if DBK2[7:5] = 110 or 111

DBK2[7:5]=0xx => Delay2=DBK2[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK2[7:5]=10x => Delay2=(64+DBK2[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK2[7:5]=110 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK2[7:5]=111 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DBK1[7:0]**: Delayed 1 Break duration

This bitfield defines the duration of the delay in Delayed 1 Break mode.

It is partially preloaded and can be updated during timer operation (new value taken into account after an update event), at the exception of the bits coding for Tdtg selection, which are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBK1[6:0] if DBK1[7] = 0
- DBK1[5:0] if DBK1[7:6] = 10
- DBK1[4:0] if DBK1[7:5] = 110 or 111

DBK1[7:5]=0xx => Delay1=DBK1[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK1[7:5]=10x => Delay1=(64+DBK1[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK1[7:5]=110 => Delay1=(32+DBK1[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK1[7:5]=111 => Delay1=(32+DBK1[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

38.7.23 TIM output override register (TIM_OOR)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOS2	OOS1N	OOS1
			rw										rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **OOC**: Output override control

This bit forces the idle state of the tim_ocx and tim_ocxn outputs to the values defined with OOSx and OOSxN bits.

0: tim_ocx and tim_ocxn outputs are in the default idle state

1: tim_ocx and tim_ocxn outputs are forced in an idle state defined with OOSx/OOSxN bits

Note: This bit can only be modified when the outputs are in idle state (MOE = 0). It is cleared when the MOE bit is set.

Bits 11:3 Reserved, must be kept at reset value.

Bit 2 **OOS2**: Output override state for output tim_oc2

This bit defines the state for output tim_oc2 when OOC = 1

0: tim_oc2 = 0

1: tim_oc2 = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **OOS1N**: Output override state for output tim_oc1n

This bit defines the state for output tim_oc1n when OOC = 1

0: tim_oc1n = 0

1: tim_oc1n = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **OOS1**: Output override state for output tim_oc1

This bit defines the state for output tim_oc1 when OOC = 1

0: tim_oc1 = 0

1: tim_oc1 = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

38.7.24 TIM DMA control register (TIM_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection
This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address).
0000: Reserved
0001: Update
0010: CC1
0011: CC2
0110: COM
0111: Trigger
Other: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length
This 5-bitfield defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address).
00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11111: 32 transfers

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address
This 5-bitfield defines the base-address for DMA transfers (when read/write access are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register.
Example:
00000: TIM_CR1
00001: TIM_CR2
00010: TIM_SMCR
...

38.7.25 TIM DMA address for full transfer (TIM_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIM_CR1 address) + (DBA + DMA index) x 4

where TIM_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM_DCR).

38.7.26 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

Table 449. TIM15 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIM_CR1**	Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value					0	0	0	0												0	0		0	0	0	0	0	0	0	0	0	0
0x04	**TIM_CR2**	Res.	TI2INV	TI1INV	ADSYNC	XORPS	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	Reset value		0	0	0	0		0															0	0	0	0	0	0	0	0	0		0
0x08	**TIM_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	SMS[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value								0			0	0			0	0									0	0	0	0		0	0	0
0x0C	**TIM_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	DIRIE	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value										0	0							0	0			0	0	0	0	0	0			0	0	0
0x10	**TIM_SR**	Res.	Res.	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	TERRF	Res.	DIRF	Res.	Res.	Res.	Res.	Res.	Res.	BGF	SBIF	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value			0	0				0	0		0							0	0			0	0		0	0	0			0	0	0
0x14	**TIM_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
	Reset value																									0	0	0			0	0	0
0x18	**TIM_CCMR1** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]		OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value							0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 449. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	**TIM_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																									0		0	0	0	0	0	0
0x24	**TIM_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIM_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIM_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	**TIM_BDTR**	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48 - 0x50	Reserved	Res.																															
0x54	**TIM_DTR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															
0x5C	**TIM_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0
0x60	**TIM_AF1**	BKCMP20E	BKCMP19E	BKCMP18E	BKCMP17E	BKCMP16E	BKCMP15E	BKCMP14E	BKCMP13E	BKCMP12E	BKCMP11E	BKCMP10E	BKCMP9E	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 449. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x64	**TIM_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	0																
0x68 - 0x08C	Reserved	Res.																															
0x090	**TIM_MPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2M[1:0]		BK1NM[1:0]		BK1M[1:0]	
	Reset value																											0	0	0	0	0	0
0x094	**TIM_MPR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x98 - 0x09C	Reserved	Res.																															
0x0A0	**TIM_OOR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOS2	OOS1N	OOS1
	Reset value																				0										0	0	0
0xA4 - 0x3D8	Reserved	Res.																															
0x3DC	**TIM_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	**TIM_DMAR**	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

38.8 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

38.8.1 TIM control register 1 (TIM_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CKD2[3:0]**: Clock division 2

This bit field indicates the division between sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_etr_in, tim_tix).

0000: No Division
0001: $t_{DTS2} = 4*t_{DTS}$
0010: $t_{DTS2} = 16*t_{DTS}$
0011: $t_{DTS2} = 64*t_{DTS}$
0100: $t_{DTS2} = 256*t_{DTS}$
0101: $t_{DTS2} = 1024*t_{DTS}$
0110: $t_{DTS2} = 4096*t_{DTS}$
0111: $t_{DTS2} = 16384*t_{DTS}$
1000: $t_{DTS2} = 65536*t_{DTS}$
1001: $t_{DTS2} = 262144*t_{DTS}$
Others: Reserved, do not program this value

Bits 23:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: $t_{DTS} = 8*t_{tim_ker_ck}$

Bit 7 **ARPE**: Auto-reload preload enable

0: TIM_ARR register is not buffered
1: TIM_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1).

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: nly counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

38.8.2 TIM control register 2 (TIM_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

0: tim_oc1n = 0 after a dead-time when MOE = 0

1: tim_oc1n = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1 = 0 after a dead-time when MOE = 0

1: tim_oc1 = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BKR register).

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when a rising edge occurs on tim_trgi (if available).

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.
Note: This bit acts only on channels that have a complementary output.

38.8.3 TIM DMA/interrupt enable register (TIM_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRIE	Res.	Res.	Res.	Res.	Res.
										rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
						rw	rw	rw		rw				rw	rw

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

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction change interrupt disabled
1: Direction change interrupt enabled

Bits 20:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

38.8.4 TIM status register (TIM_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TI1FS	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			r				rc_w0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	BGF	SBIF	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
	rc_w0	rc_w0				rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **TI1FS**: tim_ti1f status

This flag indicates the value of the tim_ti1f signal (after the digital filtering stage), for polling purpose.

0: tim_ti1f signal is low

1: tim_ti1f signal is high

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **UIOVRF**: Update Interrupt Overrun flag

This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.

0: No Update Interrupt Overrun

1: Update Interrupt Overrun

Bits 23:15 Reserved, must be kept at reset value.

Bit 14 **BGF**: Break generation flag

This flag is set by hardware as soon as a break event is forced by software, using the BG bit. It is useful to distinguish hardware-triggered from software-triggered break events. It is cleared by software by writing it to 0.

0: No break event occurred.

1: A break event was forced with the BG bit

Bit 13 **SBIF**: System break interrupt flag

This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.

This flag must be reset to re-start PWM operation.

0: No break event occurred.

1: An active level has been detected on the system break input. An interrupt is generated if BIE = 1 in the TIM_DIER register.

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected

1: The counter value has been captured in TIM_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the tim_brk input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIM_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIM_CNT matches the content of the TIM_CCR1 register. When the content of TIM_CCR1 is greater than the content of TIM_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIM_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIM_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIM_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
- At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIM_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.

38.8.5 TIM event generation register (TIM_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
								w		w				w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG:** Capture/compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIM_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

38.8.6 TIM capture/compare mode register 1 (TIM_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

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

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N =
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIM_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input.
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

38.8.7 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

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

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OC1CE**: Output Compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr input.
1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr input.

Bits 17:16, 6:4 **OC1M[4:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

00000: Frozen - The comparison between the output compare register TIM_CCR1 and the counter TIM_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

00001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00011: Toggle - tim_oc1ref toggles when TIM_CNT = TIM_CCR1.

00100: Force inactive level - tim_oc1ref is forced low.

00101: Force active level - tim_oc1ref is forced high.

00110: PWM mode 1 - Channel 1 is active as long as TIM_CNT<TIM_CCR1 else inactive.

00111: PWM mode 2 - Channel 1 is inactive as long as TIM_CNT<TIM_CCR1 else active.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIM_CCR1 disabled. TIM_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR1 enabled. Read/Write operations access the preload register. TIM_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

38.8.8 TIM capture/compare enable register (TIM_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1. Refer to the description of CC1P.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.

Bit 2 **CC1NE**: Capture/compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 450* for details.

Table 450. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CC1E bit	CC1NE bit	tim_oc1 output state	tim_oc1n output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) tim_oc1 = 0 tim_oc1n = 0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) tim_oc1 = 0	tim_oc1ref + Polarity tim_oc1n = tim_oc1ref XOR CC1NP
		0	1	0	tim_oc1ref + Polarity tim_oc1 = tim_oc1ref XOR CC1P	Output Disabled (not driven by the timer: Hi-Z) tim_oc1n = 0
		X	1	1	tim_oc1ref + Polarity + dead-time	Complementary to tim_oc1ref (not tim_oc1ref) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_oc1 = CC1P	tim_oc1ref + Polarity tim_oc1n = tim_oc1ref XOR CC1NP
		1	1	0	tim_oc1ref + Polarity tim_oc1 = tim_oc1ref XOR CC1P	Off-State (output enabled with inactive state) tim_oc1n = CC1NP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z)	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_oc1 = CC1P, tim_oc1n = CC1NP Then if the clock is present: tim_oc1 = OIS1 and tim_oc1n = OIS1N after a dead-time, assuming that OIS1 and OIS1N do not correspond to tim_oc1 and tim_oc1n both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by the GPIO controller), the OIS1, OIS1N, CC1P and CC1NP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_oc1 and tim_oc1n channels depends on the tim_oc1 and tim_oc1n channel state and GPIO control and alternate function selection registers.

38.8.9 TIM counter (TIM_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIM_ISR register. If the UIFREMAP bit in TIM_CR1 is reset, bit 31 is reserved.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

38.8.10 TIM prescaler (TIM_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (tim_cnt_ck) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM_EGR register or through trigger controller when configured in “reset mode”).

38.8.11 TIM autoreload register (TIM_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

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

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 38.4.3: Time-base unit on page 2231* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

38.8.12 TIM repetition counter register (TIM_RCR)

Address offset: 0x30

Reset value: 0x0000

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

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

– The number of PWM periods in edge-aligned mode.
– The number of half PWM period in center-aligned mode.

38.8.13 TIM capture/compare register 1 (TIM_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc1 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.
If channel CC1 is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1).
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

38.8.14 TIM break and dead-time register (TIM_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIM_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_oc1 and tim_oc1n outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_oc1 and tim_oc1n outputs are enabled if their respective enable bits are set (CC1E, CC1NE in TIM_CCER register)

See tim_oc1/tim_oc1n enable description for more details (*Section 38.8.8: TIM capture/compare enable register (TIM_CCER) on page 2351*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the tim_brk input is not active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low
1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk event) disabled
1; Break inputs (tim_brk and tim_sys_brk event) enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_oc1/tim_oc1n enable description for more details (*Section 38.8.8: TIM capture/compare enable register (TIM_CCER) on page 2351*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_oc1/tim_oc1n outputs are enabled with their inactive level as soon as CC1E = 1 or CC1NE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE = 0 on channels configured as outputs.

See tim_oc1/tim_oc1n enable description for more details (*Section 38.8.8: TIM capture/compare enable register (TIM_CCER) on page 2351*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (tim_oc1/tim_oc1n enable output signal = 0)

1: When inactive, tim_oc1/tim_oc1n outputs are forced first with their idle level as soon as CC1E = 1 or CC1NE = 1. tim_oc1/tim_oc1n enable output signal = 1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIM_BDTR register, OISx and OISxN bits in TIM_CR2 register and BKBID/BKE/BKP/AOE bits in TIM_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIM_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIM_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIM_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx → DT = DTG[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$

DTG[7:5] = 10x → DT = (64+DTG[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$

DTG[7:5] = 110 → DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$

DTG[7:5] = 111 → DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$

Example if t_{DTS}=125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

38.8.15 TIM timer deadtime register 2 (TIM_DTR2)

Address offset: 0x054

Reset value: 0x0000 0000

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

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

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

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx → DTF = DTGF[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$.

DTGF[7:5] = 10x → DTF = (64+DTGF[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$.

DTGF[7:5] = 110 → DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$.

DTGF[7:5] = 111 → DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$.

Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

38.8.16 TIM input selection register (TIM_TISEL)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

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

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIMx_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

38.8.17 TIM alternate function register 1 (TIM_AF1)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKCMP20E	BKCMP19E	BKCMP18E	BKCMP17E	BKCMP16E	BKCMP15E	BKCMP14E	BKCMP13E	BKCMP12E	BKCMP11E	BKCMP10E	BKCMP9E	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Bit 31 **BKCMP20E**: tim_brk_cmp20 enable
This bit enables the tim_brk_cmp20 for the timer's tim_brk input. tim_brk_cmp20 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp20 input disabled
1: tim_brk_cmp20 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 30 **BKCMP19E**: tim_brk_cmp19 enable

This bit enables the tim_brk_cmp19 for the timer's tim_brk input. tim_brk_cmp19 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp19 input disabled

1: tim_brk_cmp19 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 29 **BKCMP18E**: tim_brk_cmp18 enable

This bit enables the tim_brk_cmp18 for the timer's tim_brk input. tim_brk_cmp18 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp18 input disabled

1: tim_brk_cmp18 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 28 **BKCMP17E**: tim_brk_cmp17 enable

This bit enables the tim_brk_cmp17 for the timer's tim_brk input. tim_brk_cmp17 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp17 input disabled

1: tim_brk_cmp17 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 27 **BKCMP16E**: tim_brk_cmp16 enable

This bit enables the tim_brk_cmp16 for the timer's tim_brk input. tim_brk_cmp16 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp16 input disabled

1: tim_brk_cmp16 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 26 **BKCMP15E**: tim_brk_cmp15 enable

This bit enables the tim_brk_cmp15 for the timer's tim_brk input. tim_brk_cmp15 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp15 input disabled

1: tim_brk_cmp15 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 25 **BKCMP14E**: tim_brk_cmp14 enable

This bit enables the tim_brk_cmp14 for the timer's tim_brk input. tim_brk_cmp14 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp14 input disabled

1: tim_brk_cmp14 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 24 **BKCMP13E**: tim_brk_cmp13 enable

This bit enables the tim_brk_cmp13 for the timer's tim_brk input. tim_brk_cmp13 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp13 input disabled

1: tim_brk_cmp13 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 23 **BKCMP12E**: tim_brk_cmp12 enable

This bit enables the tim_brk_cmp12 for the timer's tim_brk input. tim_brk_cmp12 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp12 input disabled

1: tim_brk_cmp12 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 22 **BKCMP11E**: tim_brk_cmp11 enable

This bit enables the tim_brk_cmp11 for the timer's tim_brk input. tim_brk_cmp11 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp11 input disabled

1: tim_brk_cmp11 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 21 **BKCMP10E**: tim_brk_cmp10 enable

This bit enables the tim_brk_cmp10 for the timer's tim_brk input. tim_brk_cmp10 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp10 input disabled

1: tim_brk_cmp10 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 20 **BKCMP9E**: tim_brk_cmp9 enable

This bit enables the tim_brk_cmp9 for the timer's tim_brk input. tim_brk_cmp9 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp9 input disabled

1: tim_brk_cmp9 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 19:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input is active high

1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

38.8.18 TIM alternate function register 2 (TIM_AF2)

Address offset: 0x064

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **OCRSEL[3:0]**: tim_ocref_clr source selection
These bits select the tim_ocref_clr input source.
0000: tim_ocref_clr0
0001: tim_ocref_clr1
...
1111: tim_ocref_clr15
Refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

38.8.19 TIM multilevel protection register 1 (TIM_MPR1)

Address offset: 0x090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK1NM[1:0]		BK1M[1:0]	
												rw	rw	rw	rw

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

Bits 3:2 **BK1NM[1:0]**: Break mode for tim_oc1n output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 1:0 **BK1M[1:0]**: Break mode for tim_oc1 output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

38.8.20 TIM multilevel protection register 2 (TIM_MPR2)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **DBK2[7:0]**: Delayed 2 Break duration

This bitfield defines the duration of the delay in Delayed 2 Break mode.
It is preloaded and can be updated during timer operation (new value taken into account after an update event).
DBK2[7:5]=0xx => Delay2=DBK2[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK2[7:5]=10x => Delay2=(64+DBK2[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK2[7:5]=110 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK2[7:5]=111 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DBK1[7:0]**: Delayed 1 Break duration

This bitfield defines the duration of the delay in Delayed 1 Break mode.
It is preloaded and can be updated during timer operation (new value taken into account after an update event).
DBK2[7:5]=0xx => Delay1=DBK2[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK2[7:5]=10x => Delay1=(64+DBK2[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK2[7:5]=110 => Delay1=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK2[7:5]=111 => Delay1=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

38.8.21 TIM output override register (TIM_OOR)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOS1N	OOS1
			rw											rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **OOC**: Output override control

This bit forces the idle state of the tim_ocx and tim_ocxn outputs to the values defined with OOSx and OOSxN bits.

0: tim_ocx and tim_ocxn outputs are in the default idle state

1: tim_ocx and tim_ocxn outputs are forced in an idle state defined with OOSx/OOSxN bits

Note: This bit can only be modified when the outputs are in idle state (MOE = 0). It is cleared when the MOE bit is set.

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **OOS1N**: Output override state for output tim_oc1n

This bit defines the state for output tim_oc1n when OOC = 1

0: tim_oc1n = 0

1: tim_oc1n = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **OOS1**: Output override state for output tim_oc1

This bit defines the state for output tim_oc1 when OOC = 1

0: tim_oc1 = 0

1: tim_oc1 = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

38.8.22 TIM DMA control register (TIM_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address).

0000: Reserved

0001: Update

0010: CC1

Other: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bitfield defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bitfield defines the base-address for DMA transfers (when read/write access are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register.
Example:
00000: TIM_CR1,
00001: TIM_CR2,
00010: TIM_SMCR,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIM_CR1. In this case the transfer is done to/from 7 registers starting from the TIM_CR1 address.

38.8.23 TIM DMA address for full transfer (TIM_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIM_CR1 address) + (DBA + DMA index) x 4

where TIM_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM_DCR).

38.8.24 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 451. TIM16/TIM17 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIM_CR1**	Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value					0	0	0	0												0	0		0	0	0	0	0	0	0	0	0	0
0x04	**TIM_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
	Reset value																							0	0					0	0		0
0x08	Reserved	Res.																															
0x0C	**TIM_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
	Reset value											0												0	0	0		0				0	0
0x10	**TIM_SR**	Res.	Res.	Res.	TI1FS	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BGF	SBIF	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
	Reset value				0				0										0	0				0		0		0				0	0
0x14	**TIM_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
	Reset value																									0		0				0	0
0x18	**TIM_CCMR1** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
	TIM_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value															0	0									0	0	0	0	0	0	0	0
0x1C	Reserved	Res.																															
0x20	**TIM_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
	Reset value																													0	0	0	0
0x24	**TIM_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 451. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	**TIM_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIM_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	**TIM_BDTR**	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48 - 0x50	Reserved	Res.																															
0x54	**TIM_DTR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															
0x5C	**TIM_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x60	**TIM_AF1**	BKCMP20E	BKCMP19E	BKCMP18E	BKCMP17E	BKCMP16E	BKCMP15E	BKCMP14E	BKCMP13E	BKCMP12E	BKCMP11E	BKCMP10E	BKCMP9E	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x64	**TIM_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	0																
0x6C - 0x08C	Reserved	Res.																															
0x090	**TIM_MPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK1NM[1:0]		BK1M[1:0]	
	Reset value																													0	0	0	0
0x094	**TIM1x_MPR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x98 - 0x09C	Reserved	Res.																															

Table 451. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0A0	**TIM_OOR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOS1N	OOS1
	Reset value																				0											0	0
0xA4 - 0x3D8	Reserved	Res.																															
0x3DC	**TIM_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	**TIM_DMAR**	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

39 Low-power timer (LPTIM)

39.1 LPTIM introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. The LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

39.2 LPTIM main features

- 16-bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: configurable internal clock source (see RCC section)
  - External clock source over LPTIM input (working with no LP oscillator running, used by pulse counter application)
- 16-bit ARR autoreload register
- 16-bit capture/compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable digital glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode
- Repetition counter
- Up to 2 independent channels for:
  - Input capture
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on 10 events
- DMA request generation on the following events:
  - Update event
  - Input capture

39.3 LPTIM implementation

Table 452 describes LPTIM implementation on STM32H7Pxxx devices. The full set of features is implemented in LPTIM1, LPTIM2 and LPTIM3. LPTIM4 and LPTIM5 support a smaller set of features.

Table 452. STM32H7Pxxx LPTIM features

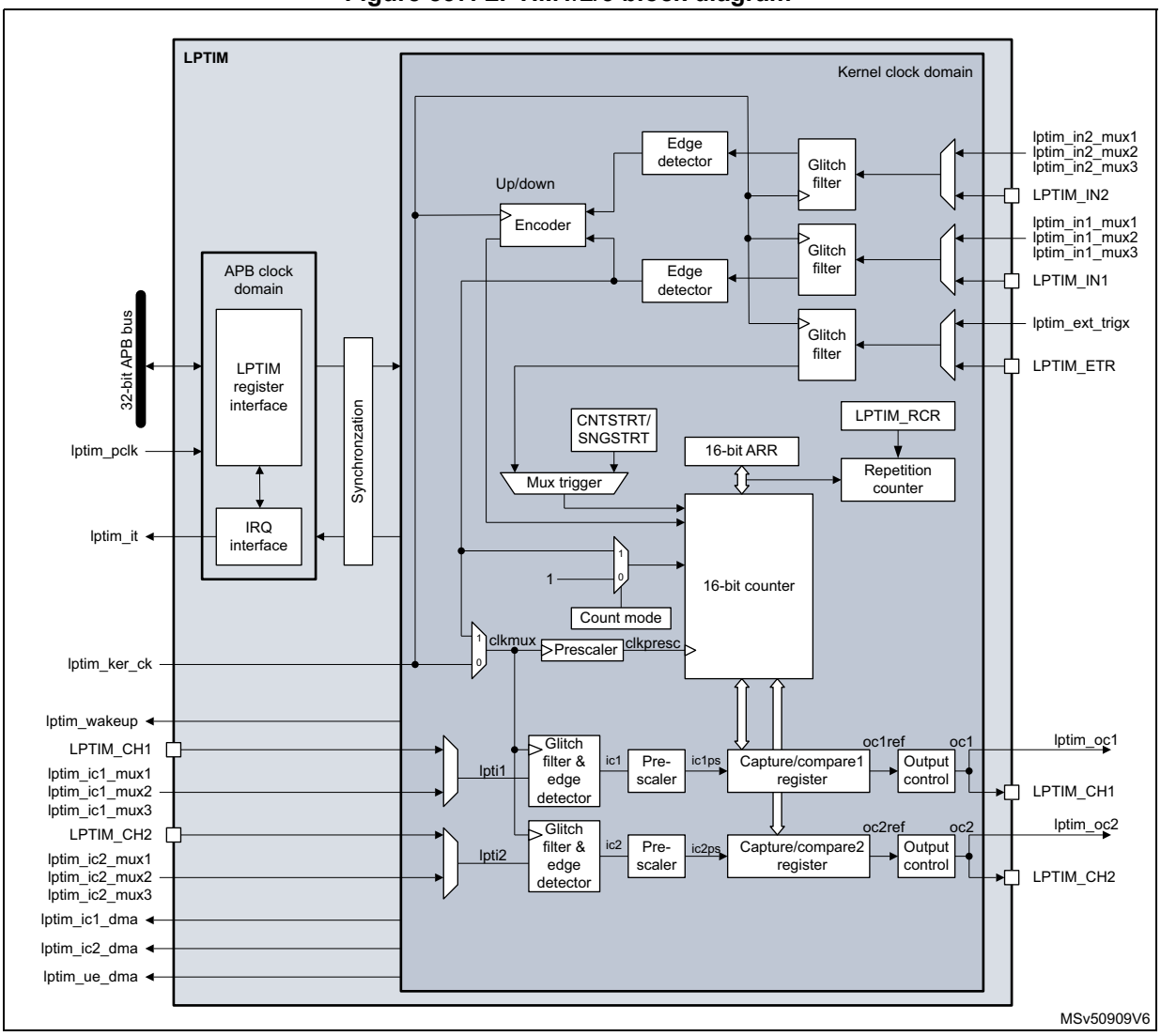
LPTIM modes/features(1)	LPTIM1	LPTIM2	LPTIM3	LPTIM4	LPTIM5
Encoder mode	X	X	X	-	-
PWM mode	X	X	X	X	X
Input Capture	X	X	X	-	-
Number of channels	2	2	2	0	0
Number of DMA requests	3	3	3	0	0
Wake-up in Stop mode	X	X	X	X	X

1. X = supported.

39.4 LPTIM functional description

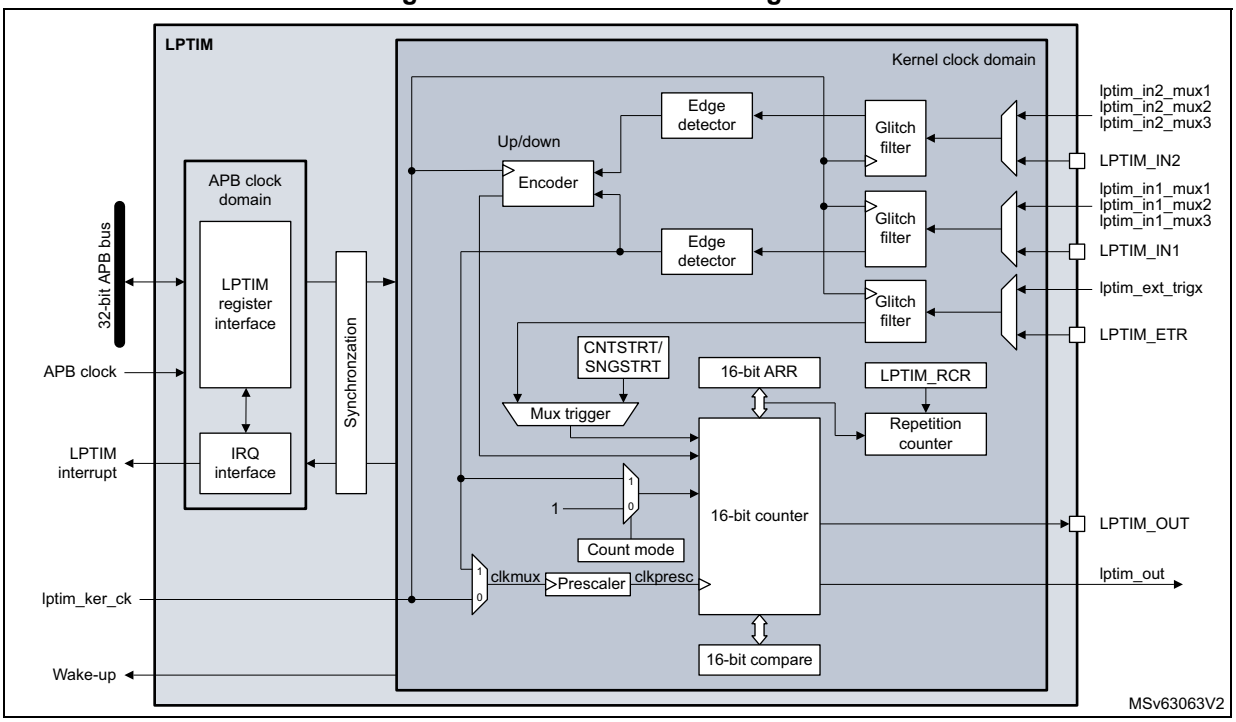
39.4.1 LPTIM block diagram

Figure 837. LPTIM1/2/3 block diagram(1)



1. Some I/Os may not be available, refer to *Section 39.4.2: LPTIM pins and internal signals*.

Figure 838. LPTIM4/5 block diagram(1)



1. Some I/Os may not be available, refer to *Section 39.4.2: LPTIM pins and internal signals*.

39.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

Table 453. LPTIM1/2/3 input/output pins

Pin name	Pin type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin on mux input 0
LPTIM_IN2	Digital input	LPTIM Input 2 from GPIO pin on mux input 0
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_CH1	Digital input/output	LPTIM channel 1 input/output GPIO pin
LPTIM_CH2	Digital input/output	LPTIM channel 2 input/output GPIO pin

Table 454. LPTIM4/5 input/output pins

Pin name	Pin type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin on mux input 0
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_OUT	Digital output	LPTIM Output GPIO pin

Table 455. LPTIM1/2/3 internal signals

Signal name	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
lptim_in2_mux1	Digital input	Internal LPTIM input 2 connected to mux input 1
lptim_in2_mux2	Digital input	Internal LPTIM input 2 connected to mux input 2
lptim_in2_mux3	Digital input	Internal LPTIM input 2 connected to mux input 3
lptim_ic1_mux1	Digital input	Internal LPTIM input capture 1 connected to mux input 1
lptim_ic1_mux2	Digital input	Internal LPTIM input capture 1 connected to mux input 2
lptim_ic1_mux3	Digital input	Internal LPTIM input capture 1 connected to mux input 3
lptim_ic2_mux1	Digital input	Internal LPTIM input capture 2 connected to mux input 1
lptim_ic2_mux2	Digital input	Internal LPTIM input capture 2 connected to mux input 2
lptim_ic2_mux3	Digital input	Internal LPTIM input capture 2 connected to mux input 3
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wake-up event
lptim_ic1_dma	Digital output	LPTIM input capture 1 DMA request
lptim_ic2_dma	Digital output	LPTIM input capture 2 DMA request
lptim_ue_dma	Digital output	LPTIM update event DMA request

Table 456. LPTIM4/5 internal signals

Signal name	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_out	Digital output	LPTIM counter output
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wake-up event

39.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter.

Table 457. LPTIM1/2/3/4/5 external trigger connections

TRIGSEL	External trigger				
	LPTIM1	LPTIM2	LPTIM3	LPTIM4	LPTIM5
lptim_ext_trig0	GPIO				
lptim_ext_trig1	rtc_alra_trg		Reserved	lptim2_ch1	lptim2_ch1
lptim_ext_trig2	rtc_alrb_trg		lptim2_ch1	Reserved	lptim3_ch1
lptim_ext_trig3	tamp_trg1		lptim4_out	lptim3_ch1	Reserved
lptim_ext_trig4	tamp_trg2		lptim5_out	lptim5_out	lptim4_out
lptim_ext_trig5	tamp_trg3		Reserved	Reserved	Reserved
lptim_ext_trig6	Reserved		Reserved	Reserved	Reserved
lptim_ext_trig7	Reserved	Reserved	Reserved	Reserved	Reserved

Table 458. LPTIM1/2/3 input 1 connections

lptim_in1_mux	LPTIM1/2/3 input 1 connected to
lptim_in1_mux0	GPIO
lptim_in1_mux1	Reserved
lptim_in1_mux2	Reserved
lptim_in1_mux3	Reserved

Table 459. LPTIM1/2/3 input 2 connections

lptim_in2_mux	LPTIM1/2 input 2 connected to
lptim_in2_mux0	GPIO
lptim_in2_mux1	Reserved
lptim_in2_mux2	Reserved
lptim_in2_mux3	Reserved

Table 460. LPTIM1/2/3 input capture 1 connections

lptim_ic1_mux	LPTIM1/2/3 input capture 1 connected to
lptim_ic1_mux0	GPIO
lptim_ic1_mux1	Reserved
lptim_ic1_mux2	Reserved
lptim_ic1_mux3	Reserved

Table 461. LPTIM1 input capture 2 connections

lptim_ic2_mux	LPTIM1 input capture 2 connected to
lptim_ic2_mux0	I/O
lptim_ic2_mux1	LSI
lptim_ic2_mux2	LSE
lptim_ic2_mux3	Reserved

Table 462. LPTIM2 input capture 2 connections

lptim_ic2_mux	LPTIM2 input capture 2 connected to
lptim_ic2_mux0	I/O
lptim_ic2_mux1	HSI/1024
lptim_ic2_mux2	CSI/128
lptim_ic2_mux3	Reserved

Table 463. LPTIM3 input capture 2 connections

lptim_ic2_mux	LPTIM3 input capture 2 connected to
lptim_ic2_mux0	I/O
lptim_ic2_mux1	Reserved
lptim_ic2_mux2	Reserved
lptim_ic2_mux3	Reserved

39.4.4 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be any configurable internal clock source selectable through the RCC (see RCC section for more details). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM can run in one of the following configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM from configurable internal clock source (see RCC section).
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM uses an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal must also be provided (first configuration). In this case, the internal clock signal frequency must be at least four times higher than the external clock signal frequency.

39.4.5 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source must first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

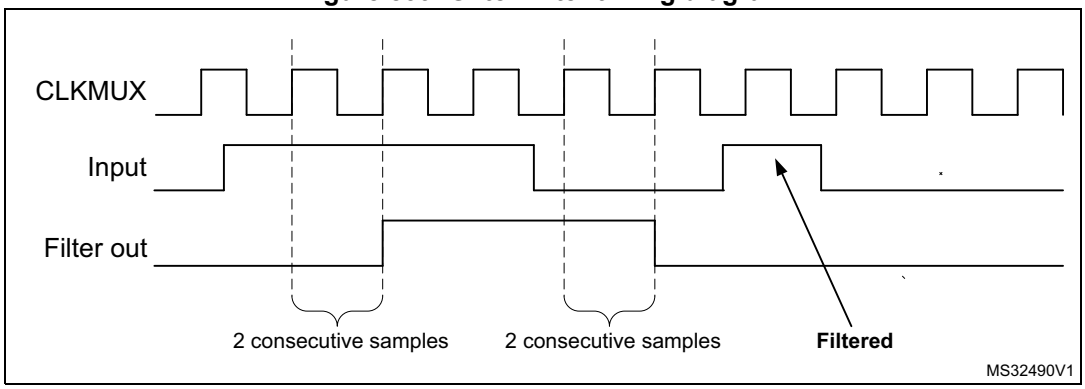
The digital filters are divided into three groups:

- The first group of digital filters protects the LPTIM internal or external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal or external trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.
- The third group of digital filters protects the LPTIM internal or external input captures. The digital filters sensitivity is controlled by the ICxF bits.

Note: The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.

The filter sensitivity acts on the number of consecutive equal samples that is detected on one of the LPTIM inputs to consider a signal level change as a valid transition. *Figure 839* shows an example of glitch filter behavior in case of a two consecutive samples programmed.

Figure 839. Glitch filter timing diagram



Note: In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT, ICxF and TRGFLT bits to 0. In this case, an external analog filter can be used to protect the LPTIM external inputs against glitches.

39.4.6 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] field. The table below lists all the possible division ratios:

Table 464. Prescaler division ratios

Programming	Dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

39.4.7 Trigger multiplexer

The LPTIM counter can be started either by software or after the detection of an active edge on one of the eight trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals 00, the LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than 00, TRIGSEL[2:0] is used to select which of the eight trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. After a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it is ignored (unless timeout function is enabled).

Note: The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled is discarded by hardware.

Note: When starting the counter by software (TRIGEN[1:0] = 00), there is a delay of 3 kernel clock cycles between the LPTIM_CR register update (set one of SNGSTRT or CNTSTRT bits) and the effective start of the counter.

39.4.8 Operating mode

The LPTIM features two operating modes:

- Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when an LPTIM update event is generated.

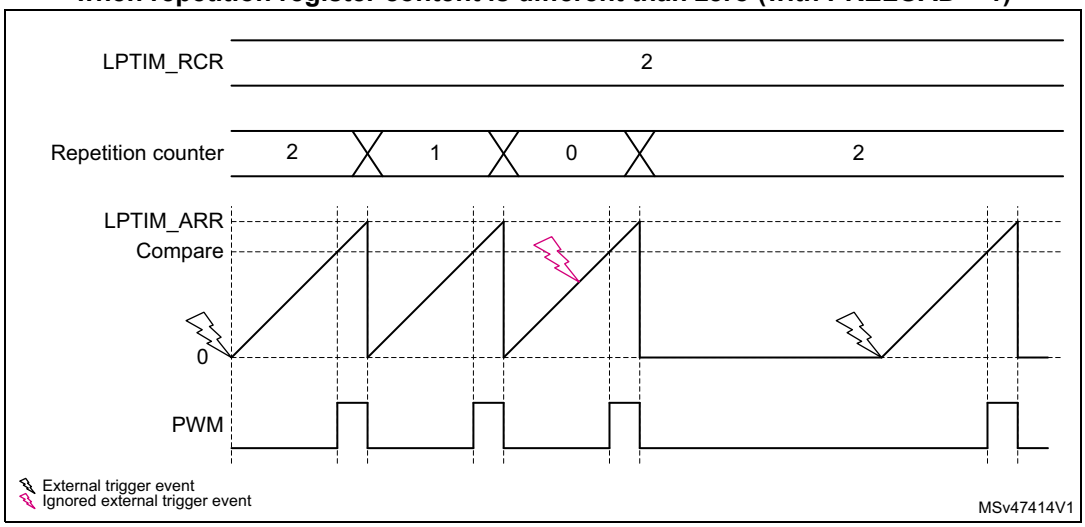
One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event re-starts the timer. Any trigger event occurring after the counter starts and before the next LPTIM update event, is discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the repetition counter has stopped (after the update event), and if the repetition register content is different from zero, the repetition counter gets reloaded with the value already contained by the repetition register and a new one-shot counting cycle is started as shown in *Figure 840*.

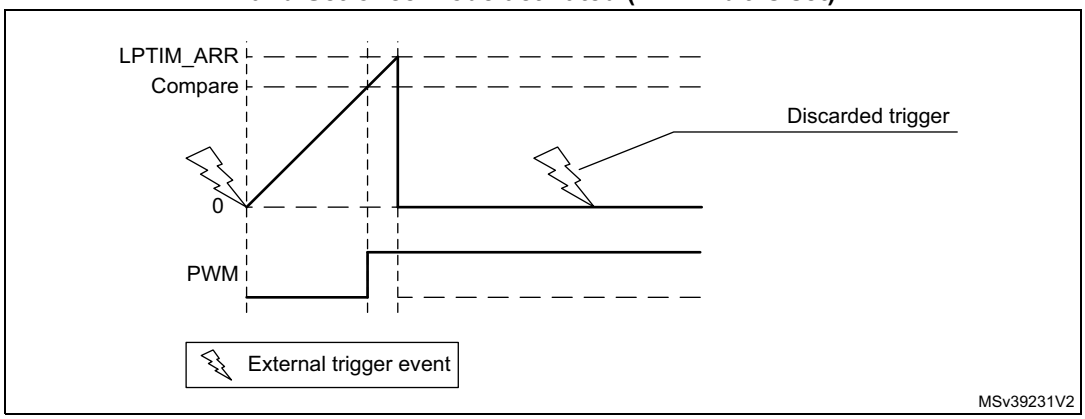
Figure 840. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1)



- Set-once mode activated:

Note that when the WAVE bitfield in the LPTIM_CFGR register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in *Figure 841*.

Figure 841. LPTIM output waveform, single-counting mode configuration and Set-once mode activated (WAVE bit is set)



In case of software start (TRIGEN[1:0] = 00), the SNGSTRT setting starts the counter for one-shot counting.

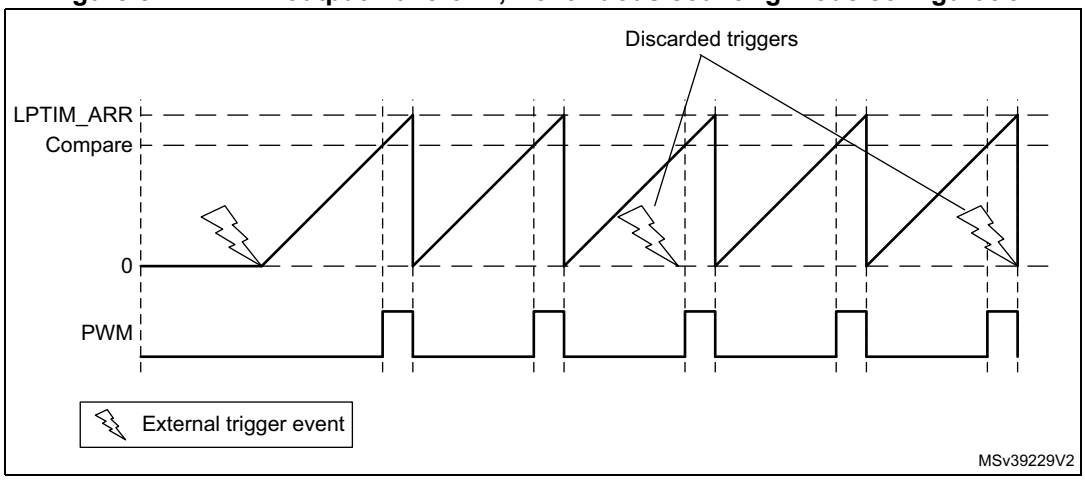
Continuous mode

To enable the continuous counting, the CNTSTRT bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set, starts the counter for continuous counting. Any subsequent external trigger event is discarded as shown in *Figure 842*.

In case of software start (TRIGEN[1:0] = 00), setting CNTSTRT starts the counter for continuous counting.

Figure 842. LPTIM output waveform, Continuous counting mode configuration



SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (ENABLE bit set to 1). It is possible to change “on the fly” from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT switches the LPTIM to the One-shot mode. The counter (if active) stops as soon as an LPTIM update event is generated.

If the One-shot mode was previously selected, setting CNTSTRT switches the LPTIM to the Continuous mode. The counter (if active) restarts as soon as it reaches ARR.

39.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMOUT bit.

The first trigger event starts the timer, any successive trigger event resets the LPTIM counter and the repetition counter and the timer restarts.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

39.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CCRx (capture/compare register), are used to generate several different waveforms on LPTIM output

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM_CNT exceeds the compare value in LPTIM_CCRx. The LPTIM output is reset as soon as a match occurs between the LPTIM_ARR and the LPTIM_CNT register. For more details see *Section 39.4.19: PWM mode*.
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require the LPTIM_ARR register value to be strictly greater than the LPTIM_CCRx register value.

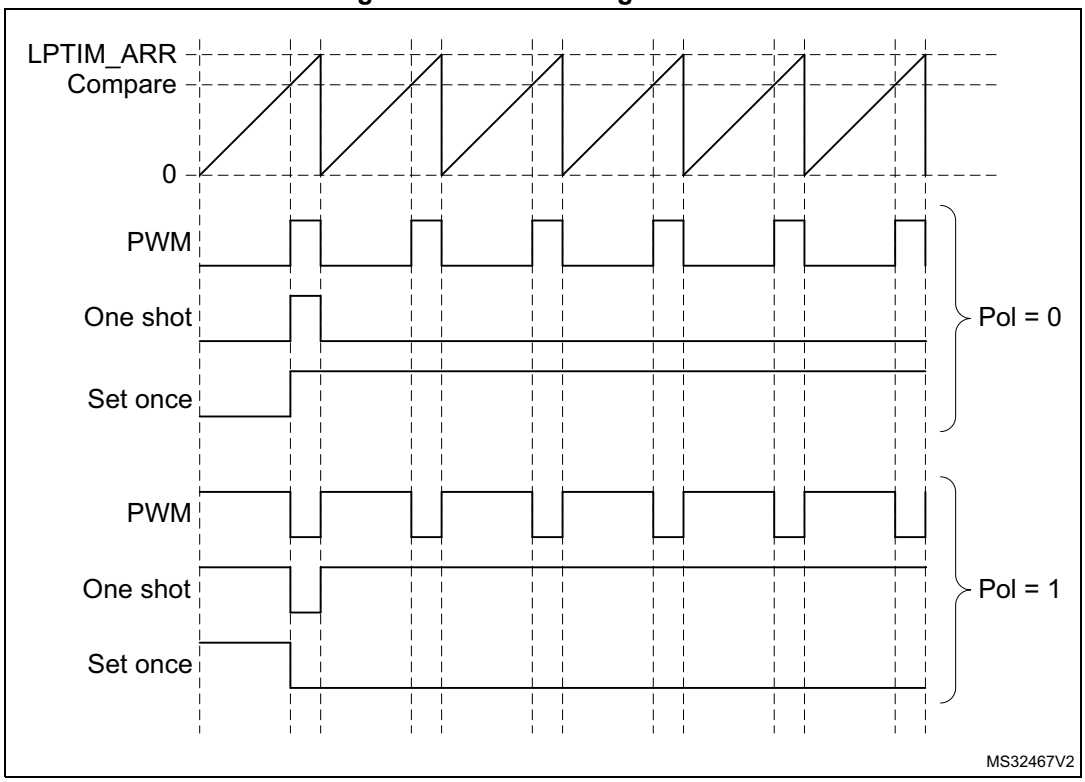
The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to 0 forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to 1 forces the LPTIM to generate a Set-once mode waveform.

The WAVPOL/CCxP bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value changes immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by two can be generated. *Figure 843* below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the WAVPOL/CCxP bit.

Figure 843. Waveform generation



39.4.11 Register update

The LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register are updated immediately after the APB bus write operation or in synchronization with the next LPTIM update event if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated:

- When the PRELOAD bit is reset to 0, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are immediately updated after any write access.
- When the PRELOAD bit is set to 1, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated at next LPTIM update event, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag, the REPOK flag and the CMPxOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register.

After a write to the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag, the REPOK flag or the CMPxOK flag be set, leads to unpredictable results.

39.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source is used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- CKSEL = 0: the LPTIM is clocked by an internal clock source
  - COUNTMODE = 0

    The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - COUNTMODE = 1

    The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.

    Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal must never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).
- CKSEL = 1: the LPTIM is clocked by an external clock source

  COUNTMODE value is don't care.

  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.

  For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.

  Since the signal injected on the LPTIM external input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

39.4.13 Timer enable

The ENABLE bit located in the LPTIM_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock cycles is needed before the LPTIM is actually enabled.

The LPTIM_CFGR register must be modified only when the LPTIM is disabled.

39.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM_CR register. After setting the COUNTRST bitfield to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic elapse before the reset is taken into account. This makes the LPTIM counter count few extra pluses between the time when the reset is triggered and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.

Note: The software ensures that COUNRST bit is '0' before generating every synchronous reset.

- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM_CR register. When this bit is set to '1', any read access to the LPTIM_CNT register resets its content to zero. Asynchronous reset must be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset must be applied only when there is enough insurance that no toggle occurs on the LPTIM Input1.
  To read reliably the content of the LPTIM_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM_CNT register.

Warning: There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. The developer must make sure that these two mechanisms are used exclusively.

39.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM_ARR must be configured before starting the counter. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by the two down and up flags in the LPTIM_ISR register. An interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to 1. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder's position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

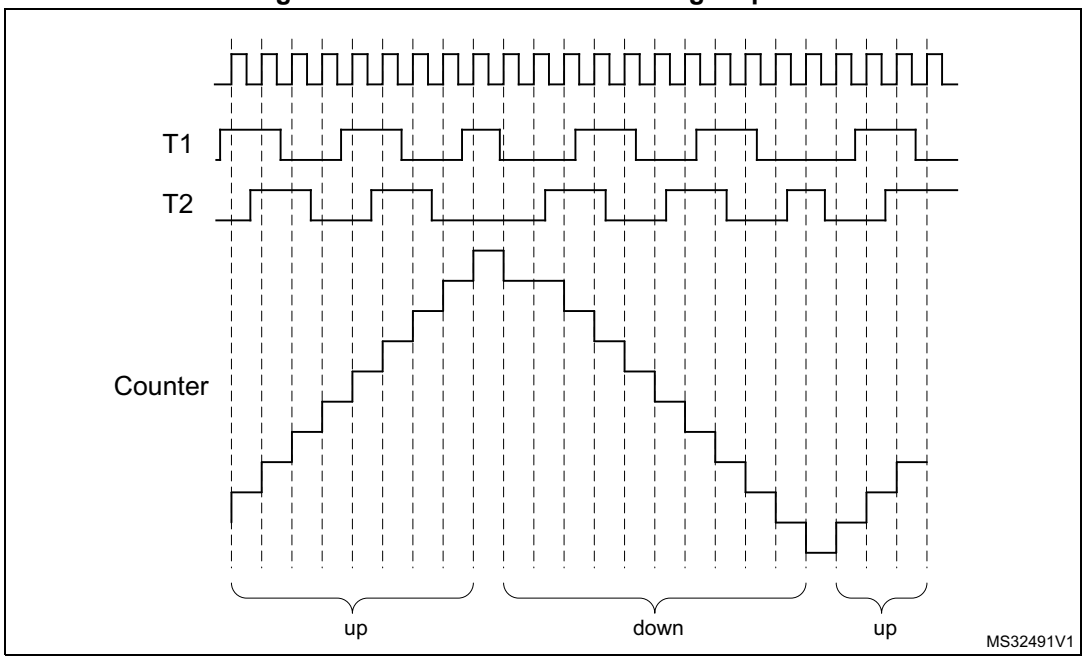
Table 465. Encoder counting scenarios

Active edge	Level on opposite signal (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
Rising Edge (Encoder sub-mode 1)	High	Down	No count	Up	No count
	Low	Up	No count	Down	No count
Falling Edge (Encoder sub-mode 2)	High	No count	Up	No count	Down
	Low	No count	Down	No count	Up
Both Edges (Encoder sub-mode 3)	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.

Caution: In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to 0. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be 000).

Figure 844. Encoder mode counting sequence



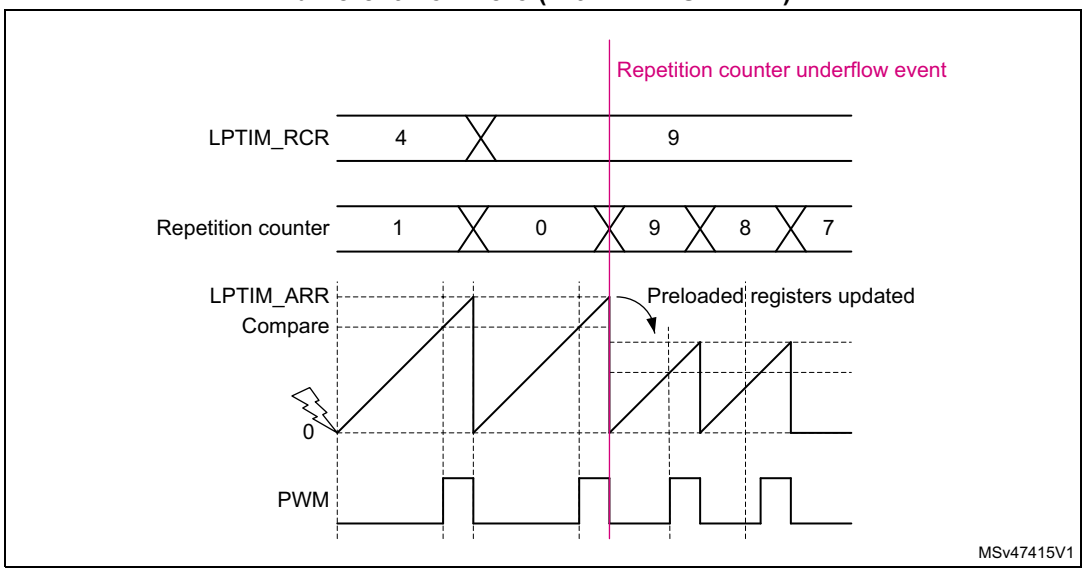
39.4.16 Repetition counter

The LPTIM features a repetition counter that decrements by 1 each time an LPTIM counter overflow event occurs. A repetition counter underflow event is generated when the repetition counter contains zero and the LPTIM counter overflows. Next to each repetition counter underflow event, the repetition counter gets loaded with the content of the REP[7:0] bitfield which belongs to the repetition register LPTIM_RCR.

A repetition underflow event is generated on each and every LPTIM counter overflow when the REP[7:0] register is set to 0.

When PRELOAD = 1, writing to the REP[7:0] bitfield has no effect on the content of the repetition counter until the next repetition underflow event occurs. The repetition counter continues to decrement each LPTIM counter overflow event and only when a repetition underflow event is generated, the new value written into REP[7:0] is loaded into the repetition counter. This behavior is depicted in *Figure 845*.

Figure 845. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1)



A repetition counter underflow event is systematically associated with LPTIM preloaded registers update (refer to *Section 39.4.11: Register update* for more information).

Repetition counter underflow event is signaled to the software through the update event (UE) flag mapped into the LPTIM_ISR register. When set, the UE flag can trigger an LPTIM interrupt if its respective update event interrupt enable (UEIE) control bit, mapped to the LPTIM_DIER register, is set.

The repetition register LPTIM_RCR is located in the APB bus interface clock domain where the repetition counter itself is located in the LPTIM kernel clock domain. Each time a new value is written to the LPTIM_RCR register, this new content is propagated from the APB bus interface clock domain to the LPTIM kernel clock domain. The new written value is then loaded to the repetition counter immediately after a repetition counter underflow event. The synchronization delay for the new written content is four APB clock cycles plus three LPTIM kernel clock cycles and it is signaled by the REPOK flag located in the LPTIM_ISR register when it is elapsed. When the LPTIM kernel clock cycle is relatively slow, for instance when the LPTIM kernel is being clocked by the LSI clock source, it can be lengthy to keep polling on the REPOK flag by software to detect that the synchronization of the LPTIM_RCR register content is finished. For that reason, the REPOK flag, when set, can generate an interrupt if its associated REPOKIE control bit in the LPTIM_DIER register is set.

Note: After a write to the LPTIM_RCR register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive writes before the REPOK flag is set, lead to unpredictable results.

Caution: When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the autoreload match event, otherwise an unpredictable behavior may occur.

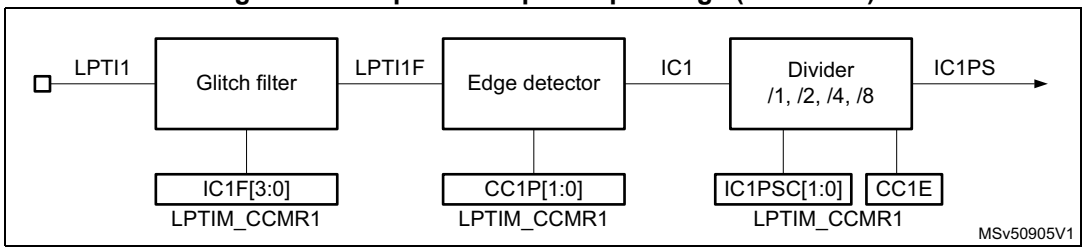
39.4.17 Capture/compare channels

Each capture/compare channel is built around a capture/compare register, an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control) for PWM.

Input stage

The input stage samples the corresponding LPTIx input to generate a filtered signal LPTIxF. Then, an edge detector with polarity selection generates ICx signal used as the capture command. It is prescaled to generate the capture command signal (ICxPS).

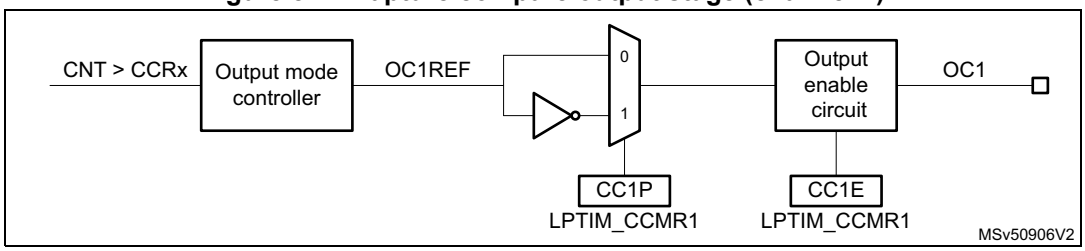
Figure 846. Capture/compare input stage (channel 1)



Output stage

The output stage generates an intermediate waveform which is then used for reference: OCxREF (active high). The polarity acts at the end of the chain.

Figure 847. Capture/compare output stage (channel 1)



39.4.18 Input capture mode

In Input capture mode, the capture/compare registers (LPTIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. Assuming input capture is enabled on a channel x (CCxE set) and when a capture occurs, the corresponding CCxIF flag (LPTIM_ISR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (LPTIM_ISR register) is set. CCxIF can be cleared by software by writing the CCxICF to 1 or by reading the captured data stored in the LPTIM_CCRx register. CCxOF is cleared by writing CCxOCF to 1.

Note: In DMA mode, the input capture channel have to be enabled (set CCxE bit) the last, after enabling the DMA request and after starting the counter. This is in order to prevent generating an input capture DMA request when the counter is not started yet.

Input capture Glitch filter latency

When a trigger event arrives on channel x input (LPTIx) and depending on the configured glitch filter (ICxF[1:0] field in CCMRx register) and on the kernel clock prescaler value (PRESC[2:0] field in CFGR register), there is a variable latency that leads to a systematic offset (see *Table 466*) between the captured value stored in the CCRx register and the real value corresponding to the capture trigger.

This offset has no impact on pulse width measurement as it is systematic and compensated between two captures.

The real capture value corresponding to the input capture trigger can be calculated using the below formula:

Real capture value = captured(LPTIM_CCRx) - offset

The relevant offset must be used depending on the glitch filter and on the kernel clock prescaler value (PRESC field in CFGR register)

Example: determining the real capture value when PRESC[2:0] = 0x2 and ICxF = 0x3.

For this configuration (PRESC[2:0] = 0x2 and ICxF = 0x3) and according to the *Table 466*, the offset is 5.

Assuming that the captured value in CCRx is 9 (LPTIM_CNT = 9), this means that the capture trigger occurred when the LPTIM_CNT was equal to 9 - 5 = 4.

Table 466. Input capture Glitch filter latency (in counter step unit)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
0	0	2
	1	7
	2	9
	3	13
1	0	3
	1	5
	2	6
	3	8
2	0	2
	1	3
	2	4
	3	5
3	0	2
	1	2
	2	3
	3	3
4	0	2
	1	2
	2	2
	3	2
5	0	2
	1	2
	2	2
	3	2

Table 466. Input capture Glitch filter latency (in counter step unit) (continued)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
6	0	2
	1	2
	2	2
	3	2
7	0	2
	1	2
	2	2
	3	2

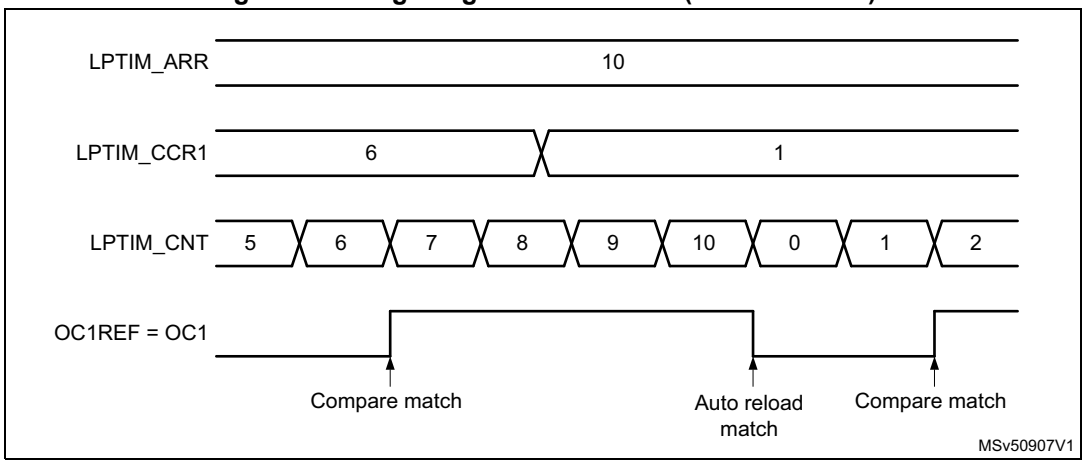
39.4.19 PWM mode

The PWM mode enables to generate a signal with a frequency determined by the value of the LPTIM_ARR register and a duty cycle determined by the value of the LPTIM_CCRx register. The LPTIM is able to generate PWM in edge-aligned mode.

OCx polarity is software programmable using the CCxP bit in the LPTIM_CCMRx register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the LPTIM_CCMRx register. Refer to the LPTIM_CCMRx register description for more details.

Figure 848 gives an example where the LPTIM channel 1 is configured in PWM mode with LPTIM_CCR1 = 6 then 1 and LPTIM_ARR=10.

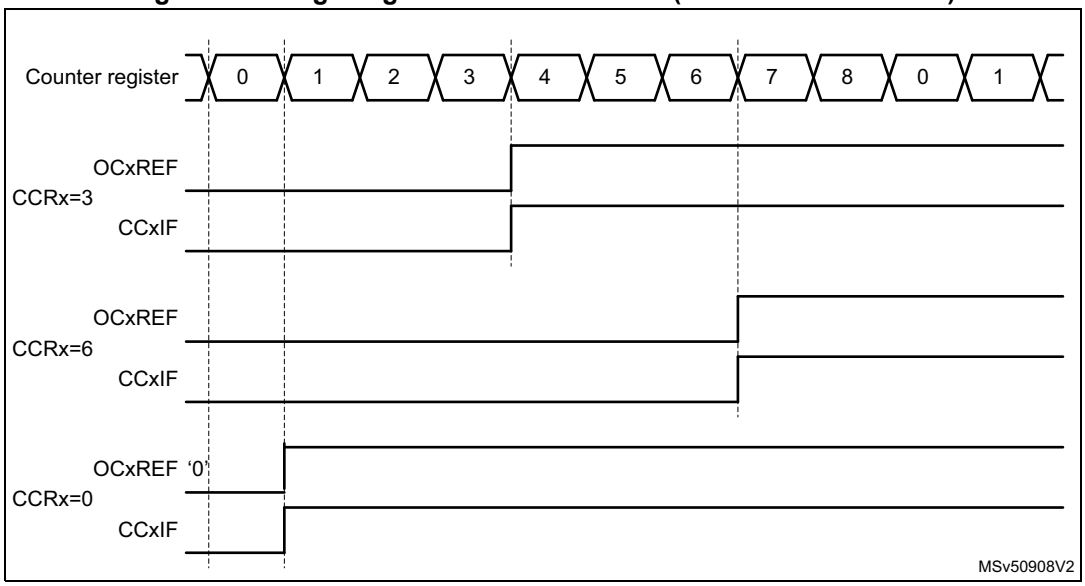
Figure 848. Edge-aligned PWM mode (PRELOAD = 1)



In the following example the reference PWM signal OCxREF is low as long as LPTIM_CNT ≤ LPTIM_CCRx else it becomes high.

Figure 849 shows some edge-aligned PWM waveforms in an example where LPTIM_ARR = 8.

Figure 849. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0)



PWM mode with immediate update PRELOAD = 0

The PWM mode with PRELOAD = 0 enables the early change of the output level within the current PWM cycle. Based on the immediate update (PRELOAD = 0) of the LPTIM_CCRx register and on the continuous comparison of LPTIM_CNT and LPTIM_CCRx registers, it permits to have a new duty cycle value applied as soon as possible within the current PWM cycle, without having to wait for the completion of the current PWM period.

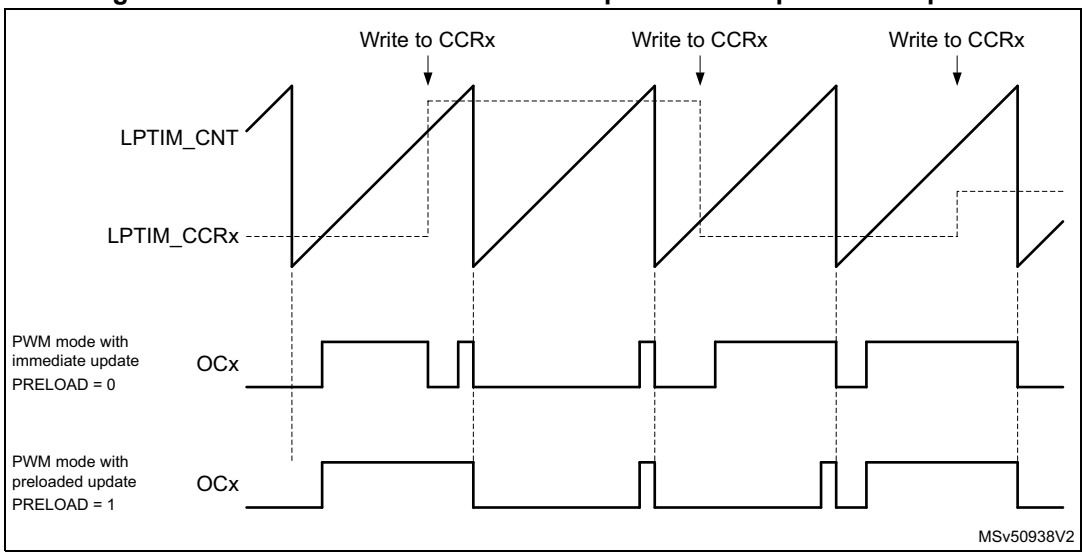
When the (PRELOAD = 0), the OCxREF signal level can be changed on-the-fly by software (or DMA) by updating the compare value in the LPTIM_CCRx register.

Depending on the written compare value and on the current counter and compare values, the OCxREF level is re-assigned as illustrated below:

- If the new compare value does not exceed the current counter value and the current compare value exceeds the counter, OCxREF level is re-assigned high as soon as the new compare value is written.
- If the new compare value exceeds the counter value and the current compare value does not exceed the counter, OCxREF level is re-assigned low as soon as the new compare value is written.

The output reference signal OCxREF level is left unchanged when none of the new compare value and the current compare value exceed the counter. *Figure 850* illustrates the behavior of the OCxREF signal level when PRELOAD = 0 and PRELOAD = 1.

Figure 850. PWM mode with immediate update versus preloaded update



Note: For both PWM modes, the compare match, auto-reload match, and the update event flags are set one LPTIM counter cycle later after the corresponding event, the OCxREF level is also changed one LPTIM counter cycle later after the corresponding event. For instance when the LPTIM_CCRx is set to 3 the CCxIF is set when the LPTIM_CNT = 4. Figure 848 illustrates this behavior.

39.4.20 DMA requests

The LPTIM can generate two categories of DMA requests:

- DMA requests used to retrieve the input-capture counter values
- DMA update requests used to re-program part of the LPTIM, multiple times, at regular intervals, without software overhead

Input capture DMA request

Each LPTIM channel has its dedicated input capture DMA request. A DMA request is generated (if CCxDE bit is set in LPTIM_DIER) and CCxIF is set each time a capture is ready in the LPTIM_CCRx register. The captured values in LPTIM_CCRx can then be transferred regularly by DMA to the desired memory destination. The CCxIF is automatically cleared by hardware when the captured value in LPTIM_CCRx register is read.

Note: The ICx DMA request signal lptim_icx_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear CCxDE bit in the LPTIM_DIER register)*
- *or if the channel x is disabled (clear CCxE bit)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*

Update event DMA request

A DMA request is generated (if UEDE is set in LPTIM_DIER) and the UE flag is set at each update event. DMA request can be used to regularly update the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx registers permitting to generate custom PWM waveforms.

The UE is automatically cleared by hardware upon any bus master (like CPU or DMA) write access to the LPTIM_ARR register.

Note: The UE DMA request signal lptim_ue_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear UEDE bit in the LPTIM_DIER register)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*
- *or if the channel x is disabled (clear CCxE bit) and all the other channels are already disabled*

39.4.21 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the timer dedicated bit configuration in the debug support (DBG) peripheral.

For further details, refer to section debug support (DBG).

39.5 LPTIM low-power modes

Note: All DMA requests must be disabled (reset UEDE and CCxDE bits) before entering Sleep, Stop and Standby modes.

39.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_DIER register:

- Compare match
- Auto-reload match (whatever the direction if Encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (Encoder mode), programmable (up / down / both).
- Update Event
- Repetition register update OK
- Input capture occurred
- Over-capture occurred
- Interrupt enable register update OK

Note: If any bit in the LPTIM_DIER register is set after that its corresponding flag in the LPTIM_ISR register (status register) is set, the interrupt is not asserted.

Table 467. Interrupt events

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop mode(1)
LPTIMx	Compare match	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Input capture(2)	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Over-capture(2)	CCxOF	CCxOIE	Write 1 to CCxOCF	Yes	Yes
	Auto-reload match	ARRM	ARRMIE	Write 1 to ARRMCF	Yes	Yes
	External trigger event	EXTTRIG	EXTTRIGIE	Write 1 to EXTTRIGCF	Yes	Yes
	Auto-reload register update OK	ARROK	ARROKIE	Write 1 to ARROKCF	Yes	Yes
	Capture/compare register update OK	CMPxOK	CMPxOKIE	Write 1 to CMPxOKCF	Yes	Yes
	Direction change to up(3)	UP	UPIE	Write 1 to UPCF	Yes	Yes
	Direction change to down(3)	DOWN	DOWNIE	Write 1 to DOWNCF	Yes	Yes
	Update event	UE	UEIE	Write 1 to UECF	Yes	Yes
	Repetition register update OK	REPOK	REPOKIE	Write 1 to REPOKCF	Yes	Yes

1. Each LPTIM event can wake up the device from Stop mode only if the LPTIM instance supports the wake-up from Stop mode feature. Refer to *Section 39.3: LPTIM implementation*.
2. If LPTIM does not implement any channel this event does not exist. Refer to *Section 39.3: LPTIM implementation*.
3. If LPTIM does not support Encoder mode feature, this event does not exist. Refer to *Section 39.3: LPTIM implementation*.

39.7 LPTIM registers

Refer to *Section 1.2: List of abbreviations for registers on page 106* for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).

39.7.1 LPTIMx interrupt and status register (LPTIMx_ISR)(x = 4 to 5)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOK	UE	DOWN	UP	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
							r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:9 Reserved, must be kept at reset value.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3: LPTIM implementation.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3: LPTIM implementation.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMP1OK**: Compare register 1 update OK

CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: Compare 1 interrupt flag

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. The CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

39.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3)

This description of the register can only be used for output compare mode. See next section for input capture mode.

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OK	Res.	Res.	Res.	Res.	CMP2 OK	Res.	Res.	Res.
							r					r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2IF	REP OK	UE	DOWN	UP	ARR OK	CMP1 OK	EXT TRIG	ARRM	CC1IF
						r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OK**: Compare register 2 update OK

CMP2OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR2 register has been successfully completed. CMP2OK flag can be cleared by writing 1 to the CMP2OKCF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Compare 2 interrupt flag

If channel CC2 is configured as output:

The CC2IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR2 register's value

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMP1OK**: Compare register 1 update OK

CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: Compare 1 interrupt flag

If channel CC1 is configured as output:

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

39.7.3 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OF	CC1OF	Res.	Res.	CC2IF	REPOK	UE	DOWN	UP	ARROK	Res.	EXTTRIG	ARRM	CC1IF
		r	r			r	r	r	r	r	r		r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OF**: Capture 2 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC2OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR2 register while CC2IF flag was already set.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 12 **CC1OF**: Capture 1 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC1OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR1 register while CC1IF flag was already set.

Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 39.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Capture 2 interrupt flag

If channel CC2 is configured as input:

CC2IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR2 register. The corresponding interrupt or DMA request is generated if enabled. The CC2OF flag is set if the CC2IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR2 register. (An edge has been detected on IC2 which matches the selected polarity). The CC2IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA). The CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. The UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: capture 1 interrupt flag

If channel CC1 is configured as input:

CC1IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR1 register. The corresponding interrupt or DMA request is generated if enabled. The CC1OF flag is set if the CC1IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR1 register. (An edge has been detected on IC1 which matches the selected polarity). The CC1IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA).CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

39.7.4 LPTIMx interrupt clear register (LPTIMx_ICR)(x = 4 to 5)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKCF	UECF	DOWNCF	UPCF	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
							w	w	w	w	w	w	w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag

Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:9 Reserved, must be kept at reset value.

Bit 8 **REPOKCF**: Repetition register update OK clear flag

Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag

Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag

Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 5 **UPCF**: Direction change to UP clear flag

Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMP1OKCF**: Compare register 1 update OK clear flag
Writing 1 to this bit clears the CMP1OK flag in the LPTIM_ISR register.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

39.7.5 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OKCF	Res.	Res.	Res.	Res.	CMP2 OKCF	Res.	Res.	Res.
							w					w			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2CF	REPOK CF	UECF	DOWN CF	UPCF	ARR OKCF	CMP1 OKCF	EXT TRIG CF	ARRM CF	CC1CF
						w	w	w	w	w	w	w	w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKCF**: Compare register 2 update OK clear flag
Writing 1 to this bit clears the CMP2OK flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMP1OKCF**: Compare register 1 update OK clear flag
Writing 1 to this bit clears the CMP1OK flag in the LPTIM_ISR register.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

39.7.6 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OCF	CC1OCF	Res.	Res.	CC2CF	REPOKCF	UECF	DOWNCF	UPCF	ARROKCF	Res.	EXTTRIGCF	ARRMCF	CC1CF
		w	w			w	w	w	w	w	w		w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OCF**: Capture/compare 2 over-capture clear flag
Writing 1 to this bit clears the CC2OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 12 **CC1OCF**: Capture/compare 1 over-capture clear flag
Writing 1 to this bit clears the CC1OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 39.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

39.7.7 LPTIMx interrupt enable register (LPTIMx_DIER)(x = 4 to 5)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 **CMP1OKIE**: Compare register 1 update OK interrupt enable
0: CMPOK register 1 interrupt disabled
1: CMPOK register 1 interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

39.7.8 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEDE	Res.	Res.	Res.	CMP2OKIE	Res.	Res.	Res.
								rw				rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 39.3.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKIE**: Compare register 2 update OK interrupt enable
0: CMPOK register 2 interrupt disabled
1: CMPOK register 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 **CMP1OKIE**: Compare register 1 update OK interrupt enable
0: CMPOK register 1 interrupt disabled
1: CMPOK register 1 interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

39.7.9 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CC2DE	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE
						rw		rw							rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OIE	CC1OIE	Res.	Res.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
		rw	rw			rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 Reserved, must be kept at reset value.

Bit 25 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled. Writing '0' to the CC2DE bit resets the associated ic2_dma_req signal.
1: CC2 DMA request enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 39.3.

Bits 22:17 Reserved, must be kept at reset value.

Bit 16 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled. Writing '0' to the CC1DE bit resets the associated ic1_dma_req signal.
1: CC1 DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 39.3.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OIE**: Capture/compare 2 over-capture interrupt enable
0: CC2 over-capture interrupt disabled
1: CC2 over-capture interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 12 **CC1OIE**: Capture/compare 1 over-capture interrupt enable
0: CC1 over-capture interrupt disabled
1: CC1 over-capture interrupt enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 39.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

39.7.10 LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNT MODE	PRE LOAD	WAV POL	WAVE	TIMOUT	TRIGEN[1:0]		Res.
							rw	rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL[2:0]			Res.	PRESC[2:0]			Res.	TRGFLT[1:0]		Res.	CKFLT[1:0]		CKPOL[1:0]		CKSEL
rw	rw	rw		rw	rw	rw		rw	rw		rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **ENC**: Encoder mode enable
The ENC bit controls the Encoder mode
0: Encoder mode disabled
1: Encoder mode enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 23 **COUNTMODE**: counter mode enabled
The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:
0: The counter is incremented following each internal clock pulse
1: The counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 **PRELOAD**: Registers update mode
The PRELOAD bit controls the LPTIM_ARR, LPTIM_RCR and the LPTIM_CCRx registers update modality
0: Registers are updated after each APB bus write access
1: Registers are updated at the end of the current LPTIM period

Bit 21 **WAVPOL**: Waveform shape polarity
The WAVPOL bit controls the output polarity
0: The LPTIM output reflects the compare results between LPTIM_CNT and LPTIM_CCRx registers
1: The LPTIM output reflects the inverse of the compare results between LPTIM_CNT and LPTIM_CCRx registers
Note: If the LPTIM implements at least one capture/compare channel, this bit is reserved. Refer to Section 39.3.

Bit 20 **WAVE**: Waveform shape
The WAVE bit controls the output shape
0: Deactivate Set-once mode
1: Activate the Set-once mode

Bit 19 **TIMOUT**: Timeout enable
The TIMOUT bit controls the Timeout feature
0: A trigger event arriving when the timer is already started is ignored
1: A trigger event arriving when the timer is already started resets and restarts the LPTIM counter and the repetition counter

Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity
The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:
00: Software trigger (counting start is initiated by software)
01: Rising edge is the active edge
10: Falling edge is the active edge
11: Both edges are active edges

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **TRIGSEL[2:0]**: Trigger selector

The TRIGSEL bits select the trigger source that serves as a trigger event for the LPTIM among the below 8 available sources:

000: lptim_ext_trig0
001: lptim_ext_trig1
010: lptim_ext_trig2
011: lptim_ext_trig3
100: lptim_ext_trig4
101: lptim_ext_trig5
110: lptim_ext_trig6
111: lptim_ext_trig7

See *Section 39.4.3: LPTIM input and trigger mapping* for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]:** Clock prescaler

The PRESC bits configure the prescaler division factor. It can be one among the following division factors:

000: /1
001: /2
010: /4
011: /8
100: /16
101: /32
110: /64
111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]:** Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: Any trigger active level change is considered as a valid trigger
01: Trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.
10: Trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.
11: Trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]:** Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: Any external clock signal level change is considered as a valid transition
01: External clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.
10: External clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.
11: External clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 2:1 **CKPOL[1:0]:** Clock polarity

When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:

00: The rising edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 1 is active.

01: The falling edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 2 is active.

10: Both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 3 is active.

11: Not allowed

Refer to *Section 39.4.15: Encoder mode* for more details about Encoder sub-modes.

Bit 0 **CKSEL:** Clock selector

The CKSEL bit selects which clock source the LPTIM uses:

0: LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)

1: LPTIM is clocked by an external clock source through the LPTIM external Input1

Caution: The LPTIM_CFGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to 0).

39.7.11 LPTIM control register (LPTIM_CR)

Address offset: 0x010

Reset value: 0x0000 0000

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

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

Bit 4 **RSTARE:** Reset after read enable

This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM_CNT register asynchronously resets LPTIM_CNT register content.
This bit can be set only when the LPTIM is enabled.

Bit 3 **COUNTRST:** Counter reset

This bit is set by software and cleared by hardware. When set to '1' this bit triggers a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock can be different from APB clock).
This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Caution: COUNTRST must never be set to '1' by software before it is already cleared to '0' by hardware. Software must consequently check that COUNTRST bit is already cleared to '0' before attempting to set it to '1'.

Bit 2 **CNTSTRT:** Timer start in Continuous mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = 00), setting this bit starts the LPTIM in Continuous mode.

If the software start is disabled (TRIGEN[1:0] different than 00), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.

If this bit is set when a single pulse mode counting is ongoing, then the timer does not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode.

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Bit 1 **SNGSTRT:** LPTIM start in Single mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = 00), setting this bit starts the LPTIM in single pulse mode.

If the software start is disabled (TRIGEN[1:0] different than 00), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.

If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM stops at the following match between LPTIM_ARR and LPTIM_CNT registers.

This bit can only be set when the LPTIM is enabled. It is automatically reset by hardware.

Bit 0 **ENABLE:** LPTIM enable

The ENABLE bit is set and cleared by software.

0: LPTIM is disabled. Writing '0' to the ENABLE bit resets all the DMA request signals (input capture and update event DMA requests).

1: LPTIM is enabled

39.7.12 LPTIM compare register 1 (LPTIM_CCR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR1[15:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the capture/compare 1 register.

Depending on the PRELOAD option, the CCR1 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 becomes read-only, it contains the counter value transferred by the last input capture 1 event. The LPTIM_CCR1 register is read-only and cannot be programmed.

If LPTIM does not implement any channel:

The compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on LPTIM output.

Caution: The LPTIM_CCR1 register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

39.7.13 LPTIM autoreload register (LPTIM_ARR)

Address offset: 0x018

Reset value: 0x0000 0001

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ARR[15:0]:** Auto reload value
ARR is the autoreload value for the LPTIM.
This value must be strictly greater than the CCRx[15:0] value.

Caution: The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

39.7.14 LPTIM counter register (LPTIM_CNT)

Address offset: 0x01C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]:** Counter value
When the LPTIM is running, reading the LPTIM_CNT register may return unreliable values. In this case it is necessary to perform consecutive reads until two returned values are identical.

39.7.15 LPTIM configuration register 2 (LPTIM_CFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]	
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
										rw	rw			rw	rw

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

Bits 21:20 **IC2SEL[1:0]**: LPTIM input capture 2 selection

The IC2SEL bits control the LPTIM Input capture 2 multiplexer, which connects LPTIM Input capture 2 to one of the available inputs.
00: lptim_ic2_mux0
01: lptim_ic2_mux1
10: lptim_ic2_mux2
11: lptim_ic2_mux3
For connection details refer to *Section 39.4.3: LPTIM input and trigger mapping*.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **IC1SEL[1:0]**: LPTIM input capture 1 selection

The IC1SEL bits control the LPTIM Input capture 1 multiplexer, which connects LPTIM Input capture 1 to one of the available inputs.
00: lptim_ic1_mux0
01: lptim_ic1_mux1
10: lptim_ic1_mux2
11: lptim_ic1_mux3
For connection details refer to *Section 39.4.3: LPTIM input and trigger mapping*.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **IN2SEL[1:0]**: LPTIM input 2 selection

The IN2SEL bits control the LPTIM input 2 multiplexer, which connects LPTIM input 2 to one of the available inputs.
00: lptim_in2_mux0
01: lptim_in2_mux1
10: lptim_in2_mux2
11: lptim_in2_mux3
For connection details refer to *Section 39.4.3: LPTIM input and trigger mapping*.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **IN1SEL[1:0]**: LPTIM input 1 selection

The IN1SEL bits control the LPTIM input 1 multiplexer, which connects LPTIM input 1 to one of the available inputs.
00: lptim_in1_mux0
01: lptim_in1_mux1
10: lptim_in1_mux2
11: lptim_in1_mux3
For connection details refer to *Section 39.4.3: LPTIM input and trigger mapping*.

39.7.16 LPTIM repetition register (LPTIM_RCR)

Address offset: 0x028

Reset value: 0x0000 0000

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

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

Bits 7:0 **REP[7:0]**: Repetition register value
REP is the repetition value for the LPTIM.

Caution: The LPTIM_RCR register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the auto reload match event, otherwise an unpredictable behavior may occur.

39.7.17 LPTIM capture/compare mode register 1 (LPTIM_CCMR1)

Address offset: 0x02C

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (PWM mode). The direction of a channel is defined by configuring the corresponding CCxSEL bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2 SEL
		rw	rw			rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1 SEL
		rw	rw			rw	rw					rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC2F[1:0]**: Input capture 2 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: Any external input capture signal level change is considered as a valid transition

01: External input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **IC2PSC[1:0]**: Input capture 2 prescaler

This bitfield defines the ratio of the prescaler acting on the CC2 input (IC2).

00: No prescaler, capture is done each time an edge is detected on the capture input

01: Capture is done once every 2 events

10: Capture is done once every 4 events

11: Capture is done once every 8 events

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

Bits 19:18 **CC2P[1:0]**: Capture/compare 2 output polarity.

Condition: CC2 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC2 active high

1: OC2 active low

Condition: CC2 as input

This field is used to select the IC2 polarity for capture operations.

00: Rising edge, circuit is sensitive to IC2 rising edge

01: Falling edge, circuit is sensitive to IC2 falling edge

10: Reserved, do not use this configuration.

11: Both edges, circuit is sensitive to both IC2 rising and falling edges.

Bit 17 **CC2E**: Capture/compare 2 output enable.

Condition: CC2 as output

0: Off - OC2 is not active. Writing '0' to the CC2E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC2 signal is output on the corresponding output pin

Condition: CC2 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 2 (LPTIM_CCR2) or not.

0: Capture disabled. Writing '0' to the CC2E bit resets the associated ic2_dma_req signal.

1: Capture enabled.

Bit 16 **CC2SEL**: Capture/compare 2 selection

This bitfield defines the direction of the channel, input (capture) or output mode.

0: CC2 channel is configured in output PWM mode

1: CC2 channel is configured in input capture mode

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

Bits 13:12 **IC1F[1:0]**: Input capture 1 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: Any external input capture signal level change is considered as a valid transition

01: External input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).

00: No prescaler, capture is done each time an edge is detected on the capture input

01: Capture is done once every 2 events

10: Capture is done once every 4 events

11: Capture is done once every 8 events

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 **CC1P[1:0]**: Capture/compare 1 output polarity.

Condition: CC1 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC1 active high, the LPTIM output reflects the compare results between LPTIM_ARR and LPTIM_CCRx registers

1: OC1 active low, the LPTIM output reflects the inverse of the compare results between LPTIM_ARR and LPTIM_CCRx registers

Condition: CC1 as input

This field is used to select the IC1 polarity for capture operations.

00: Rising edge, circuit is sensitive to IC1 rising edge

01: Falling edge, circuit is sensitive to IC1 falling edge

10: Reserved, do not use this configuration.

11: Both edges, circuit is sensitive to both IC1 rising and falling edges.

Bit 1 **CC1E**: Capture/compare 1 output enable.

Condition: CC1 as output

0: Off - OC1 is not active. Writing '0' to the CC1E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC1 signal is output on the corresponding output pin

Condition: CC1 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (LPTIM_CCR1) or not.

0: Capture disabled. Writing '0' to the CC1E bit resets the associated ic1_dma_req signal.

1: Capture enabled.

Bit 0 **CC1SEL**: Capture/compare 1 selection

This bitfield defines the direction of the channel input (capture) or output mode.

0: CC1 channel is configured in output PWM mode

1: CC1 channel is configured in input capture mode

Caution: After a write to the LPTIM_CCMRx register, a new write operation to the same register can only be performed after a delay that must be equal or greater than the value of (PRESC × 3)

kernel clock cycles, PRESC[2:0] being the clock decimal division factor (1, 2, 4,..128). Any successive write violating this delay, leads to unpredictable results.

Caution: The CCxSEL, ICxF[1:0], CCxP[1:0] and ICxPSC[1:0] fields must only be modified when the channel x is disabled (CCxE bit reset to 0).

If LPTIM does not implement any channel this register is reserved. Refer to Section 39.3.

39.7.18 LPTIM compare register 2 (LPTIM_CCR2)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR2[15:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the capture/compare 2 register.

Depending on the PRELOAD option, the CCR2 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 2 contains the value to be compared to the counter LPTIM_CNT and signaled on OC2 output.

If channel CC2 is configured as input:

CCR2 becomes read-only, it contains the counter value transferred by the last input capture 2 event. The LPTIM_CCR2 register is read-only and cannot be programmed.

Caution: The LPTIM_CCR2 register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

Note: If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 39.3: LPTIM implementation*.

39.7.19 LPTIM register map

The following table summarizes the LPTIM registers.

Table 468. LPTIM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LPTIMx_ISR (x = 4 to 5)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOK	UE	DOWN(2)	UP(2)	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
	Reset value								0																0	0	0	0	0	0	0	0	0

Table 468. LPTIM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LPTIMx_ISR (x = 1 to 3) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	CMP2OK[1]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IF[1]	REPOK	UE	DOWN[2]	UP[2]	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_ISR (x = 1 to 3) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF[1]	CC1OF	Res.	Res.	CC2IF[1]	REPOK	UE	DOWN[2]	UP[2]	ARROK	Res.	EXTTRIG	ARRM	CC1IF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x004	LPTIMx_ICR (x = 4 to 5)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKCF	UECF	DOWNCF[2]	UPCF[2]	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0															0	0	0	0	0	0	0	0	0	0
0x004	LPTIMx_ICR (x = 1 to 3) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	CMP2OKCF[1]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2CF[1]	REPOKCF	UECF	DOWNCF[2]	UPCF[2]	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_ICR (x = 1 to 3) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OCF[1]	CC1OCF	Res.	Res.	CC2CF[1]	REPOKCF	UECF	DOWNCF[2]	UPCF[2]	ARROKCF	Res.	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x008	LPTIMx_DIER (x = 4 to 5)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKIE	UEIE	DOWNIE[2]	UPIE[2]	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x008	LPTIMx_DIER (x = 1 to 3) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEDE	Res.	Res.	Res.	CMP2OKIE[1]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IE[1]	REPOKIE	UEIE	DOWNIE[2]	UPIE[2]	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
	Reset value									0				0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_DIER (x = 1 to 3) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	CC2DE[1]	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	Res.	Res.	CC2OIE[1]	CC1OIE	Res.	Res.	CC2IE[1]	REPOKIE	UEIE	DOWNIE[2]	UPIE[2]	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
	Reset value							0		0							0			0	0			0	0	0	0	0	0		0	0	0
0x00C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC[2]	COUNTMODE	PRELOAD	WAVPOL[3]	WAVE	TIMOUT	TRIGEN		Res.	TRIGSEL[2:0]		Res.	PRESC			Res.	TRGFLT		CKFLT		CKPOL		CKSEL		
	Reset value								0	0	0	0	0	0	0	0		0	0	0		0	0	0	0	0			0	0	0	0	0	0
0x010	LPTIM_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
	Reset value																												0	0	0	0	0
0x014	LPTIM_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 468. LPTIM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	LPTIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x01C	LPTIM_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	LPTIM_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
	Reset value											0	0			0	0											0	0			0	0
0x028	LPTIM_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x02C	LPTIM_CCMR1(4)	Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2SEL	Res.	Res.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1SEL
	Reset value			0	0			0	0					0	0	0	0			0	0			0	0					0	0	0	0
0x034	LPTIM_CCR2(5)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. If LPTIM does not implement at least 2 channels this bit is reserved. Refer to *Section 39.3: LPTIM implementation*.
2. If LPTIM does not support Encoder mode feature, this bit is reserved. Refer to *Section 39.3: LPTIM implementation*.
3. If the LPTIM implements at least one capture/compare channel, this bit is reserved. Refer to *Section 39.3: LPTIM implementation*.
4. If LPTIM does not implement any channel this register is reserved. Refer to *Section 39.3: LPTIM implementation*.
5. If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 39.3: LPTIM implementation*.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

40 Independent watchdog (IWDG)

40.1 IWDG introduction

The independent watchdog (IWDG) peripheral offers a high safety level, thanks to its capability to detect malfunctions due to software or hardware failures.

The IWDG is clocked by an independent clock, and stays active even if the main clock fails.

In addition, the watchdog function is performed in the V_{DD} voltage domain, allowing the IWDG to remain functional even in low power modes. Refer to *Section 40.3* to check the capability of the IWDG in this product.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, making it very reliable to detect any unexpected behavior.

40.2 IWDG main features

- 12-bit down-counter
- Dual voltage domain, thus enabling operation in low power modes
- Independent clock
- Early wake-up interrupt generation
- Reset generation
  - In case of timeout
  - In case of refresh outside the expected window

40.3 IWDG implementation

Table 469. IWDG features (1)

IWDG modes/features	IWDG
LSI used as IWDG kernel clock (iwdg_ker_ck)	X
Window function	X
Early wake-up interrupt generation	X
System reset generation(2)	X
Capability to work in system Stop	X
Capability to work in system Standby	X
Capability to generate an interrupt in system Stop	X
Capability to generate an interrupt in system Standby	-
Capability to be frozen when the microcontroller enters in Debug mode(3)	X
Option bytes to control the activity in Stop mode(4)	X
Option bytes to control the activity in Standby mode(5)	X
Option bytes to control the Hardware mode(6)	X

1. ‘X’ = supported, ‘-’ = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via DBG_IWDG_STOP in DBG section.

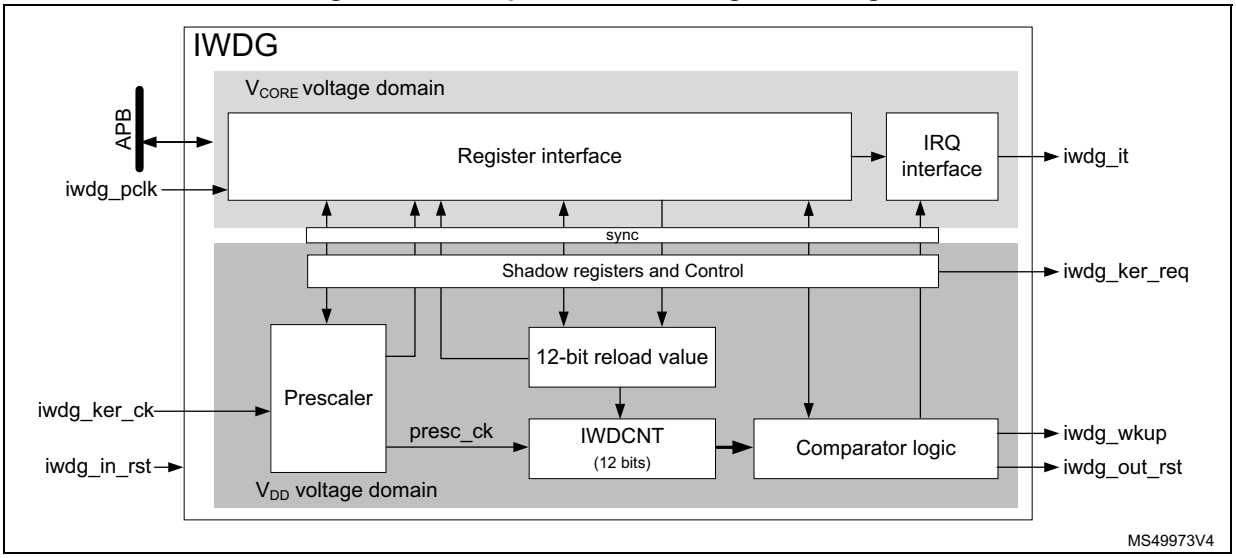
4. Controlled via the option byte IWDG_STOP in FLASH section.
5. Controlled via the option byte IWDG_STDBY in FLASH section.
6. Controlled via the option byte IWDG_SW in FLASH section.

40.4 IWDG functional description

40.4.1 IWDG block diagram

Figure 851 shows the functional blocks of the independent watchdog module.

Figure 851. Independent watchdog block diagram



The register and IRQ interfaces are located into the V_{CORE} voltage domain. The watchdog function itself is located into the V_{DD} voltage domain to remain functional in low power modes. See *Section 40.3* for IWDG capabilities.

The register and IRQ interfaces are mainly clocked by the APB clock (iwdg_pclk), while the watchdog function is clocked by a dedicated kernel clock (iwdg_ker_ck). A synchronization mechanism makes the data exchange between the two domains possible. Note that most of the registers located in the register interface are shadowed into the V_{DD} voltage domain.

The IWDG down-counter (IWDCNT) is clocked by the prescaled clock (presc_ck). The prescaled clock is generated from the kernel clock iwdg_ker_ck divided by the prescaler, according to PR[3:0] bitfield.

The table below gives the timing delays according to the actions performed on the IWDG: changing the prescaler, the timeout, the window or the early wake-up comparator values or doing a refresh.

Table 470. IWDG delays versus actions (1)

TD_{RVU}, TD_{PVU}		TD_{WVU}		TD_{EWU}		$TD_{Refresh}$		$TD_{RefAuto}$	
Min	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
5 T_k	6 T_k	-	6 T_k	-	6 T_k	2 T_k	2 T_k + T_p	-	T_p

1. Tk represents a period of the kernel clock input, Tp represents a period of presc_ck.

40.4.2 IWDG internal signals

The list of IWDG internal signals is detailed in *Table 471*.

Table 471. IWDG internal input/output signals

Signal name	Signal type	Description
iwdg_ker_ck	Input	IWDG kernel clock
iwdg_ker_req	Input	IWDG kernel clock request
iwdg_pclk	Input	IWDG APB clock
iwdg_out_rst	Output	IWDG reset output
iwdg_in_rst	Input	IWDG reset input
iwdg_wkup	Output	IWDG wake-up event
iwdg_it	Output	IWDG early wake-up interrupt

40.4.3 Software and hardware watchdog modes

The watchdog modes allow the application to select the way the IWDG is enabled, either by software commands (Software watchdog mode), or automatically (Hardware watchdog mode). All other functions work similarly for both Software and Hardware modes.

The software watchdog mode is the default working mode. The independent watchdog is started by writing the value 0x0000 CCCC into the *IWDG key register (IWDG_KR)*, and the IWDCNT starts counting down from the reset value (0xFFF).

In the hardware watchdog mode the independent watchdog is generally started automatically at power-on, or every time it is reset (via iwdg_in_rst). The IWDCNT down-counter starts counting down from the reset value 0xFFF. The hardware watchdog mode feature is enabled through the device option bits, see *Section 40.3* for details.

When the IWDG is enabled the ONF flag is set to 1.

When the IWDCNT reaches 0x000, a reset signal is generated (iwdg_out_rst asserted).

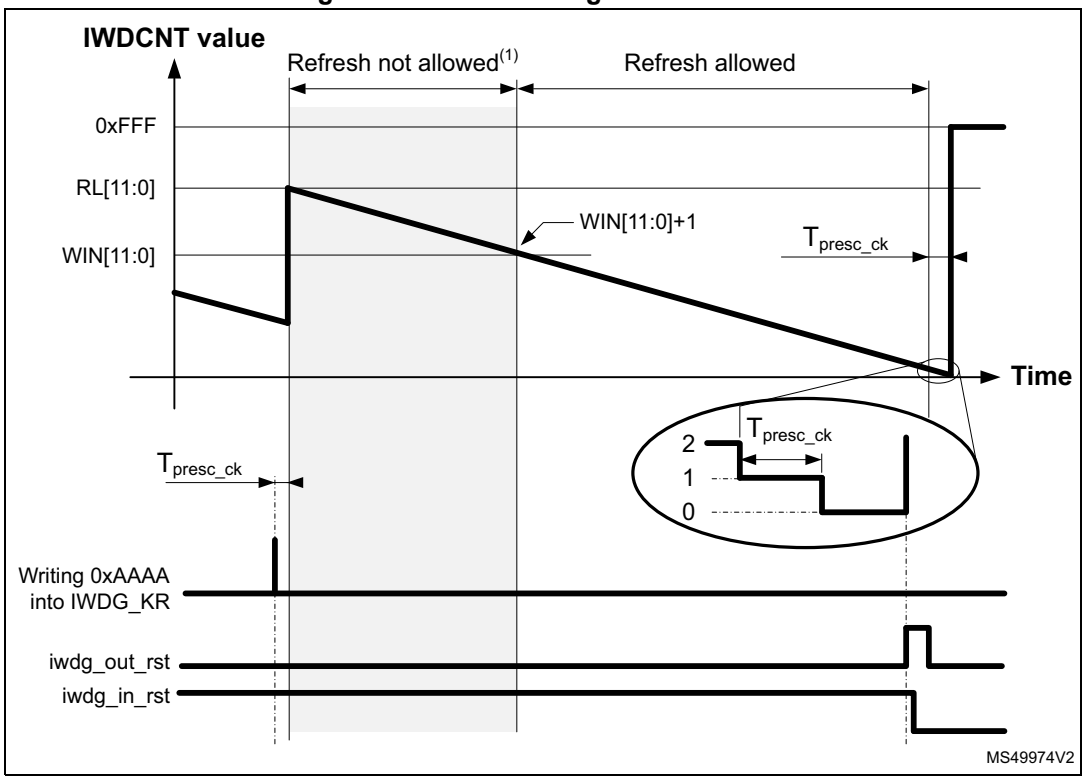
Whenever the key value 0x0000 AAAA is written in the *IWDG key register (IWDG_KR)*, the IWDG_RLR value is reloaded into the IWDCNT, and the watchdog reset is prevented.

Due to re-synchronization delays, the IWDG must be refreshed before the IWDCNT down-counter reaches 1.

Once started, the IWDG can be stopped only when it is reset (iwdg_in_rst asserted).

As shown in *Figure 852*, when the refresh command is executed, one period of presc_ck later, the IWDCNT is reloaded with the content of RL[11:0].

Figure 852. Reset timing due to timeout



1. If window option activated.

If the IWDG is not refreshed before the IWDCNT reaches 1, the IWDG generates a reset (iwdg_out_rst is asserted). In return, the RCC resets the IWDG (assertion of iwdg_in_rst) to clear the reset source.

Note: The iwdg_out_rst signal typically triggers a system reset, however, this behavior may vary across products. For more details, refer to Chapter 40.3.

40.4.4 Window option

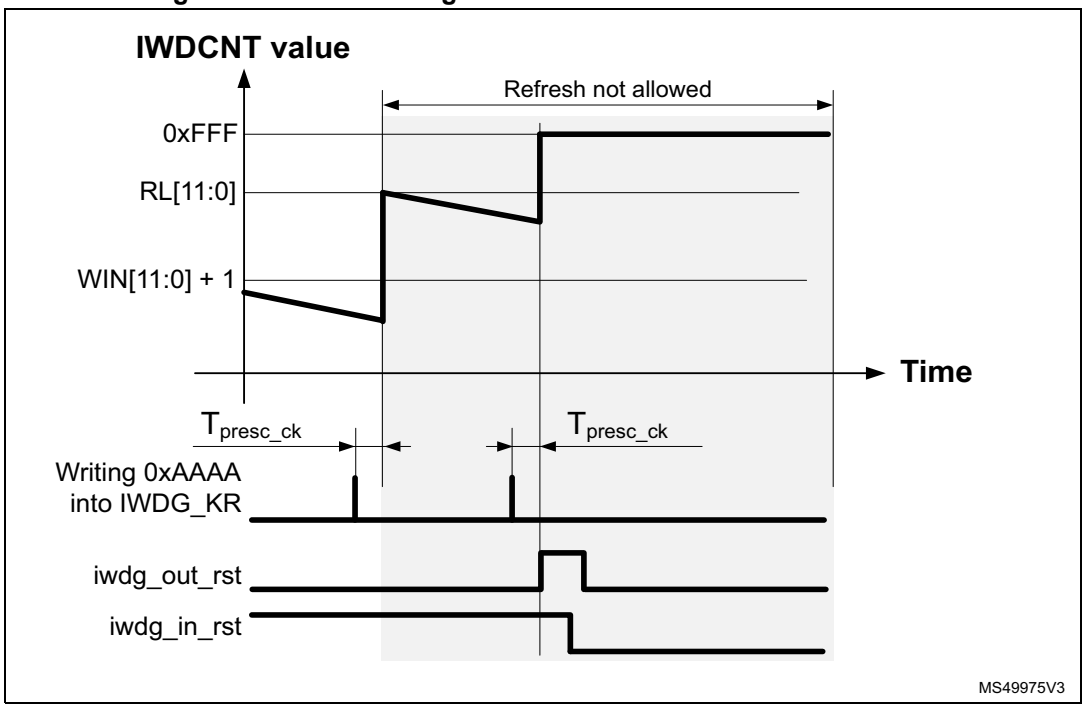
The IWDG can also work as a window watchdog, by setting the appropriate window in the *IWDG window register (IWDG_WINR)*.

If the reload operation is performed while the counter is greater than WIN[11:0] + 1, a reset is generated. WIN[11:0] is located in the *IWDG window register (IWDG_WINR)*. As shown in *Figure 853*, the reset is generated one period of presc_ck after the unexpected refresh command.

The default value of the *IWDG window register (IWDG_WINR)* is 0x0000 0FFF, so, if not updated, the window option is disabled.

As soon as the window value changes, the down-counter (IWDCNT) is reloaded with the RL[11:0] value, to ease the estimation for where the next refresh must take place.

Figure 853. Reset timing due to refresh in the not allowed area



Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the IWDG prescaler by programming *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Write to the *IWDG window register (IWDG_WINR)*. This automatically reloads the IWDCNT down-counter with the RL[11:0] value.
7. Wait for the registers to be updated (IWDG_SR = 0x0000 0100).
8. Write 0x0000 0000 into *IWDG key register (IWDG_KR)* to write-protect registers.

Note: Step 7 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

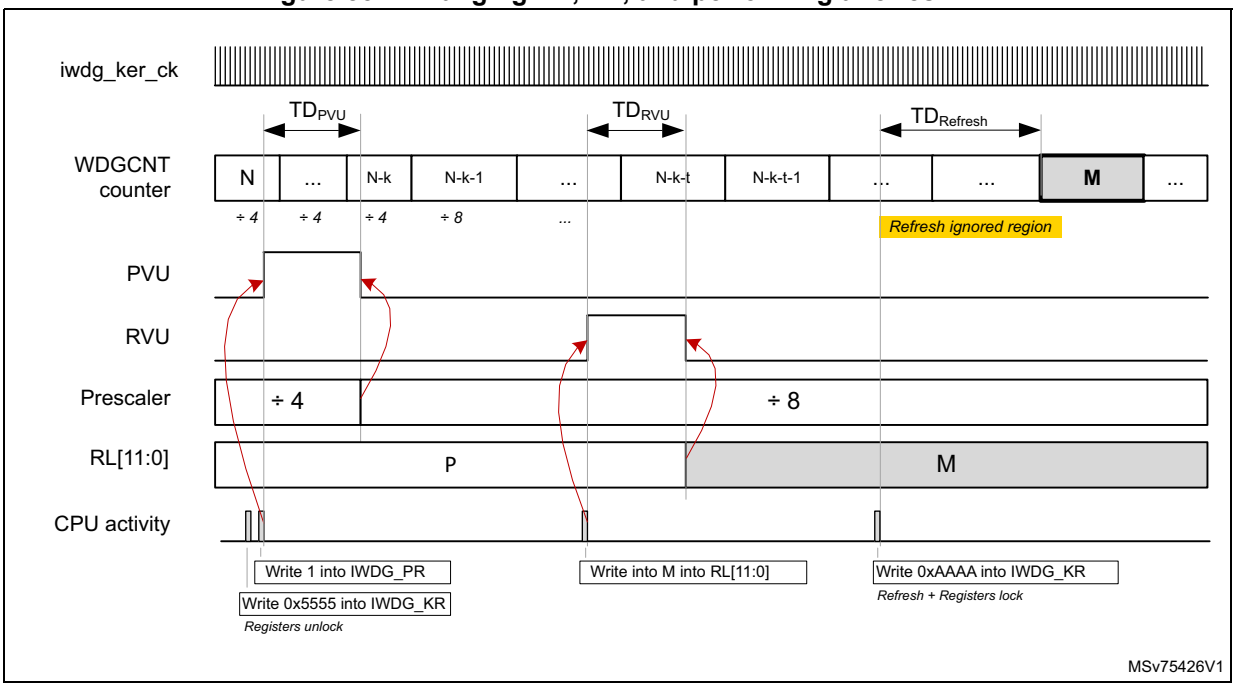
Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the prescaler by programming the *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Wait for the registers to be updated (IWDG_SR = 0x0000 0100).
7. Refresh the counter with RL[11:0] value, and write-protect registers by writing 0x0000 AAAA into *IWDG key register (IWDG_KR)*.

The figure below shows a sequence example changing the prescaler, the reload value, and then performing a refresh.

Figure 854. Changing PR, RL, and performing a refresh(1)



1. Refer to *Table 470: IWDG delays versus actions* for details on timing values.

Note: When the new prescaler value is accepted by the IWDG (falling edge of PVU), the new division ratio is effective when the current division sequence is completed (N-k in the drawing).

Note: The timeout delay between the refresh (write 0xAAAA into IWDG_KR) and the watchdog reset is increased by TDRefresh.

If a refresh command is sent while the previous refresh command is not yet completed, this last refresh command is ignored.

Updating the window comparator

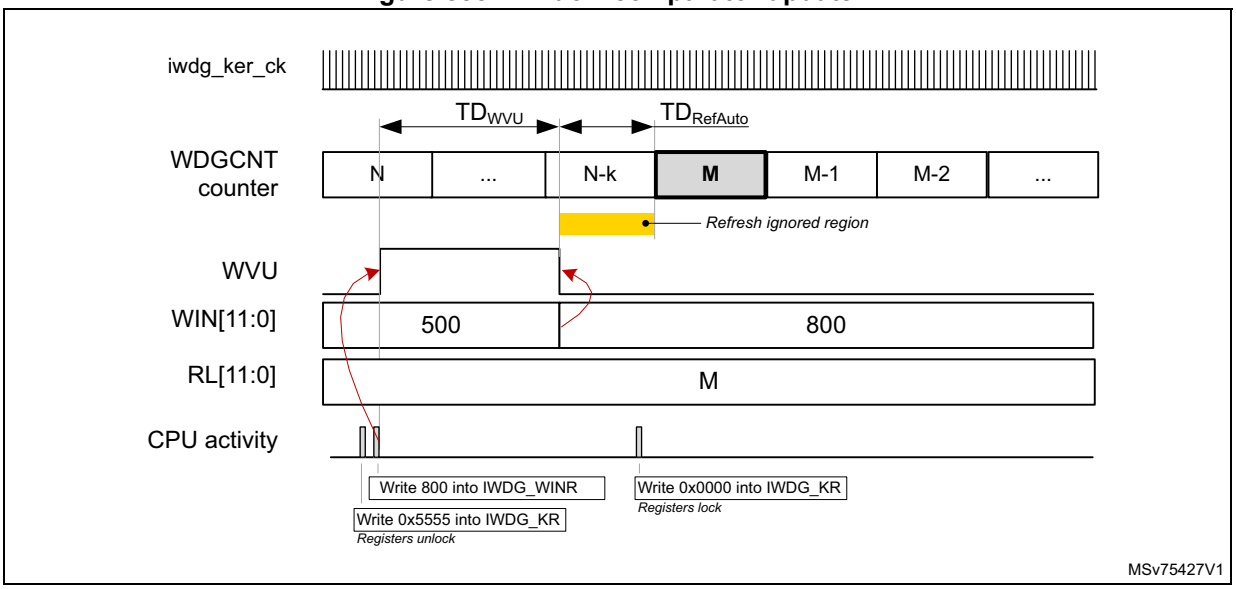
It is possible to update the window comparator when the IWDG is already running. The IWDCNT is reloaded as well. The following sequence can be performed to update the window comparator:

1. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
2. Write to the IWDG window register (IWDG_WINR). This automatically reloads the IWDCNT down-counter with RL[11:0] value.
3. Wait for WVU = 0
4. Lock registers by writing IWDG_KR to 0x0000 0000

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Figure 855 shows this sequence. As soon as the IWDG_WINR register is written, the WVU flag goes high for TDwvu. After a window comparator update, a refresh is automatically performed. The refresh is effective in the worst case, on the next prescaler clock active edge after the falling edge of WVU (TDRefAuto)

Figure 855. Window comparator update(1)



1. Refer to *Table 470: IWDG delays versus actions* for details on timing values.

40.4.5 Debug

When the processor enters into Debug mode (core halted), the IWDCNT down-counter either continues to work normally or stops, depending on debug capability of the product. Refer to *Section 40.3* for details on the capabilities of this product.

40.4.6 Register access protection

Write accesses to *IWDG prescaler register (IWDG_PR)*, *IWDG reload register (IWDG_RLR)*, *IWDG early wake-up interrupt register (IWDG_EWCR)* and *IWDG window register (IWDG_WINR)* are protected. To modify them, first write 0x0000 5555 in the *IWDG key register (IWDG_KR)*. A write access to this register with a different value breaks the

sequence and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value or the window value is ongoing.

40.5 IWDG low power modes

Depending on option bytes configuration, the IWDG can continue counting or not during the low power modes. Refer to *Section 40.3* for details.

Table 472. Effect of low power modes on IWDG

Mode	Description
Sleep	No effect. IWDG interrupts cause the device to exit from the mode.
Stop	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 40.3* for details. IWDG interrupts cause the device to exit the Stop mode.
Standby	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 40.3* for details. IWDG interrupts do not make the device to exit from Standby mode.

40.6 IWDG interrupts

The IWDG offers the possibility to generate an early interrupt depending on the value of the down-counter. The early interrupt is enabled by setting the EWIE bit of the *IWDG early wake-up interrupt register (IWDG_EWCR)* to 1.

A comparator value (EWIT[11:0]) allows the application to define the position where the early interrupt must be generated.

When the IWDCNT down-counter reaches the value of EWIT[11:0] - 1, the iwdg_wkup is activated, making it possible for the system to exit from low power modes, if needed.

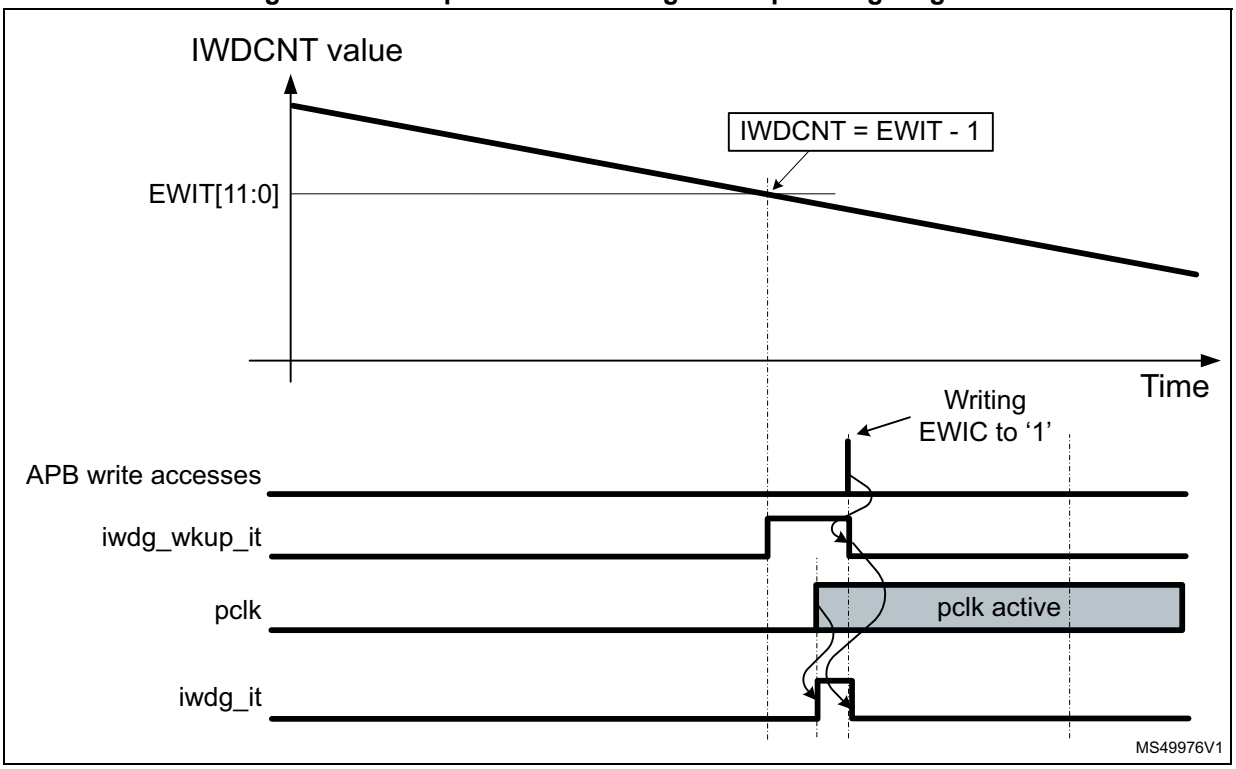
When the APB clock is available, the iwdg_it is activated as well.

In addition, the flag EWIF of the *IWDG status register (IWDG_SR)* is set to 1.

The EWI interrupt is acknowledged by writing 1 to the EWIC bit in the *IWDG early wake-up interrupt register (IWDG_EWCR)*.

Writing into the IWDG_EWCR register also triggers a refresh of the down-counter (IWDCNT) with the reload value RL[11:0].

Figure 856. Independent watchdog interrupt timing diagram



The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the watchdog reset is generated.

Changing the early wake-up comparator value

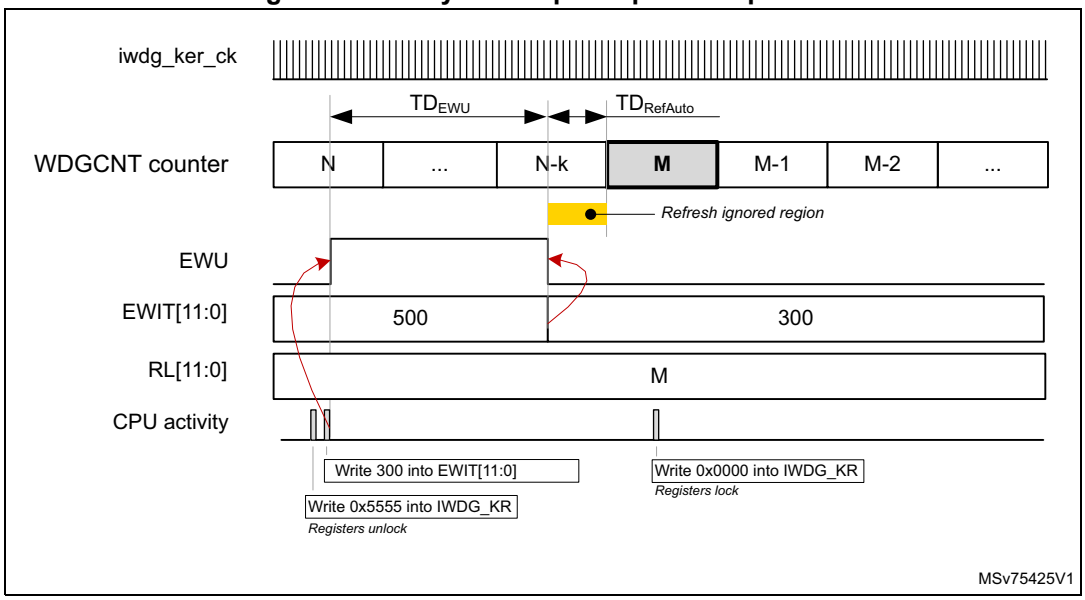
It is possible to change the early wake-up comparator value or to enable/disable the interrupt generation at any time, by performing the following sequence:

1. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
2. Enable or disable the early wake-up interrupt, and/or program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
3. Wait for EWU = 0, EWU is located into the *IWDG status register (IWDG_SR)*.
4. Write-protect registers by writing 0x0000 0000 to *IWDG key register (IWDG_KR)*.

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Figure 857 shows this sequence. During the early wake-up comparator update operation the flag EWU remains to 1 for TDewu. A refresh is automatically performed. The refresh is effective on the worst case, on the next prescaler clock active edge after the falling edge of EWU (TDRefAuto).

Figure 857. Early wake-up comparator update(1)



1. Refer to *Table 470: IWDG delays versus actions* for details on timing values.

Table 473 summarizes the IWDG interrupt request.

Table 473. IWDG interrupt request

Interrupt event	Event flag	Interrupt clear method	Interrupt enable control bit	Activated interrupt	
				iwdg_it	iwdg_wkup_it
IWDCNT reaches EWIT value	EWIF	Writing EWIC to 1	EWIE	Y(1)	Y(2)

1. Generated when a clock is present on iwdg_pclk input.
2. Generated when a clock is present on iwdg_ker_ck input.

40.7 IWDG registers

Refer to *Section 1.2 on page 106* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

Most of the registers located into the register interface are shadowed into the V_{DD} voltage domain. When the iwdg_in_rst is asserted, the watchdog logic and the shadow registers located into the V_{DD} voltage domain are reset.

When the application reads back a watchdog register, the hardware transfers the value of the corresponding shadow register to the register interface.

When the application writes a watchdog register, the hardware updates the corresponding shadow register.

40.7.1 IWDG key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0x0000)

These bits can be used for several functions, depending upon the value written by the application:

- 0xAAAA: reloads the RL[11:0] value into the IWDCNT down-counter (watchdog refresh), and write-protects registers. This value must be written by software at regular intervals, otherwise the watchdog generates a reset when the counter reaches 0.
- 0x5555: enables write-accesses to the registers.
- 0xCCCC: enables the watchdog (except if the hardware watchdog option is selected) and write-protects registers.
- values different from 0x5555: write-protects registers.

Note that only IWDG_PR, IWDG_RLR, IWDG_EWCR and IWDG_WINR registers have a write-protection mechanism.

40.7.2 IWDG prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

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

Bits 3:0 **PR[3:0]:** Prescaler divider

These bits are write access protected, see *Section 40.4.6*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the *IWDG status register (IWDG_SR)* must be reset to be able to change the prescaler divider.

0000: divider / 4
0001: divider / 8
0010: divider / 16
0011: divider / 32
0100: divider / 64
0101: divider / 128
0110: divider / 256
0111: divider / 512
Others: divider / 1024

Note: Reading this register returns the prescaler value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG status register (IWDG_SR) is reset.

40.7.3 IWDG reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF

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

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected, see *Section 40.4.6*. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the *IWDG key register (IWDG_KR)*. The watchdog counter counts down from this value. The timeout period is a function of this value and the prescaler.clock. It is not recommended to set RL[11:0] to a value lower than 2.

The RVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.

40.7.4 IWDG status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (0xFFFF FEFF)

This register contains various status flags. Note that the mask value between parenthesis means that the reset value of ONF bit is not defined. When the IWDG is configured in

software mode, the reset value of ONF bit is 0, when the IWDG is configured in hardware mode, the reset value of ONF bit is 1.

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

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **EWIF**: Watchdog early interrupt flag

This bit is set to ‘1’ by hardware in order to indicate that an early interrupt is pending. This bit must be cleared by the software by writing the bit EWIC of IWDG_EWCR register to ‘1’.

Bits 13:9 Reserved, must be kept at reset value.

Bit 8 **ONF**: Watchdog enable status bit

Set to ‘1’ by hardware as soon as the IWDG is started. In software mode, it remains to '1' until the IWDG is reset. In hardware mode, this bit is always set to '1'.

0: The IWDG is not activated

1: The IWDG is activated and needs to be refreshed regularly by the application

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **EWU**: Watchdog interrupt comparator value update

This bit is set by hardware to indicate that an update of the interrupt comparator value (EWIT[11:0]) or an update of the EWIE is ongoing. It is reset by hardware when the update operation is completed in the V_{DD} voltage domain. Refer to *Table 470: IWDG delays versus actions* for delay values.

The EWIT[11:0] and EWIE fields can be updated only when EWU bit is reset.

Bit 2 **WVU:** Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 470: IWDG delays versus actions* for delay values.

The window value can be updated only when WVU bit is reset.

This bit is generated only if generic “window” = 1.

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 470: IWDG delays versus actions* for delay values.

The reload value can be updated only when RVU bit is reset.

Bit 0 **PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain. Refer to *Table 470: IWDG delays versus actions* for delay values.

The prescaler value can be updated only when PVU bit is reset.

Note: If several reload, prescaler, early interrupt position or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, to wait until WVU bit is reset before changing the window value, and to wait until EWU bit is reset before changing the

early interrupt position value. After updating the prescaler and/or the reload/window/early interrupt value, it is not necessary to wait until RVU or PVU or WVU or EWU is reset before continuing code execution, except in case of low power mode entry.

40.7.5 IWDG window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]:** Watchdog counter window value

These bits are write access protected, see *Section 40.4.6*.They contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the IWDCNT downcounter must be reloaded when its value is lower than WIN[11:0] + 1 and greater than 1.

The WVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the IWDG status register (IWDG_SR) is reset.

40.7.6 IWDG early wake-up interrupt register (IWDG_EWCR)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIE**: Watchdog early interrupt enable

Set and reset by software.

0: The early interrupt interface is disabled.

1: The early interrupt interface is enabled.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the value of this bit.

Bit 14 **EWIC**: Watchdog early interrupt acknowledge

The software must write a 1 into this bit in order to acknowledge the early wake-up interrupt and to clear the EWIF flag. Writing 0 has not effect, reading this flag returns a 0.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:0 **EWIT[11:0]:** Watchdog counter early wake-up interrupt value

These bits are write access protected (see *Section 40.4.6*). They are written by software to define at which position of the IWDCNT down-counter the early wake-up interrupt must be generated. The early interrupt is generated when the IWDCNT is lower or equal to EWIT[11:0] - 1.

EWIT[11:0] must be bigger than 1.

An interrupt is generated only if EWIE = 1.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the Early wake-up comparator value and the Interrupt enable bit from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the EWU bit in the IWDG status register (IWDG_SR) is reset.

40.7.7 IWDG register map

Table 474. IWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	IWDG_KR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	IWDG_PR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[3:0]			
	Reset value																													0	0	0	0
0x08	IWDG_RLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	IWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF	Res.	Res.	Res.	Res.	Res.	ONF	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
	Reset value																		0						x					0	0	0	0
0x10	IWDG_WINR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WIN[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x14	IWDG_EWCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIE	EWIC	Res.	Res.	EWIT[11:0]											
	Reset value																	0	0			0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

41 System window watchdog (WWDG)

41.1 WWDG introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence.

The watchdog circuit generates a reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit is cleared. A reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter reaches the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications requiring the watchdog to react within an accurate timing window.

41.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see *Figure 859*)
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40

41.3 WWDG implementation

Table 475. WWDG features(1)

WWDG mode / feature	WWDG
Window function	X
Early wake-up interrupt generation	X
System reset generation(2)	X
Capability to work in system Stop	-
Capability to work in system Standby	-
Capability to be frozen when the microcontroller enters in Debug mode(3)	X
Option bytes to control the hardware mode	-

1. “X” = supported, “-” = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via DBG_WWDG_STOP in DBG block.

41.4 WWDG functional description

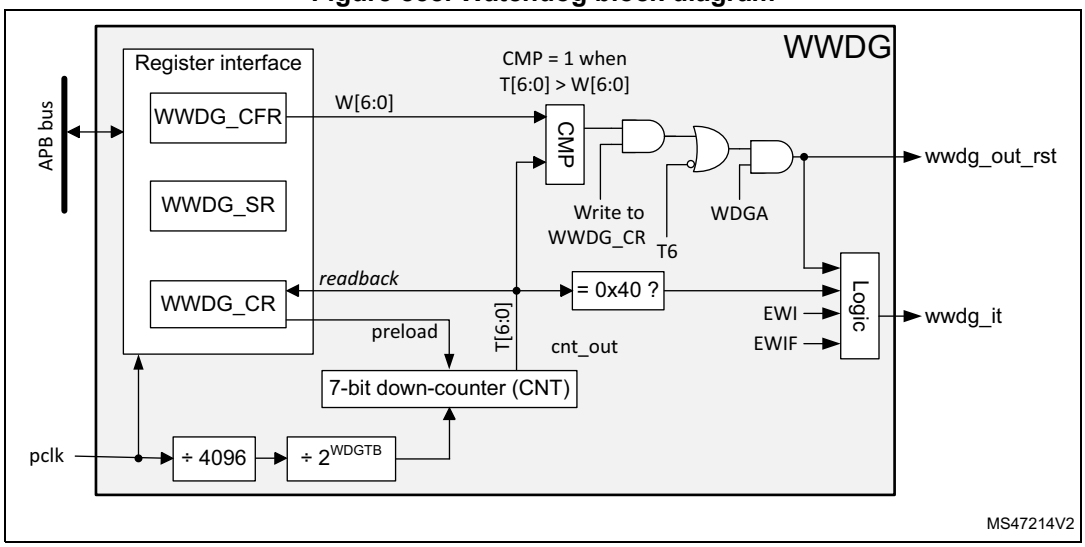
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register), and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent a reset. This operation can take place only when the counter value is lower than or equal to the window register value, and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to *Figure 858* and to *Section 41.4.2: WWDG internal signals* for the WWDG block diagram.

41.4.1 WWDG block diagram

Figure 858. Watchdog block diagram



41.4.2 WWDG internal signals

Table 476 gives the list of WWDG internal signals.

Table 476. WWDG internal input/output signals

Signal name	Signal type	Description
pclk	Digital input	APB bus clock
wwdg_out_rst	Digital output	WWDG reset signal output
wwdg_it	Digital output	WWDG early interrupt output

41.4.3 Enabling the watchdog

When the user option WWDG_SW selects “Software window watchdog”, the watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again, except by a reset.

When the user option WWDG_SW selects “Hardware window watchdog”, the watchdog is always enabled after a reset, it cannot be disabled.

Note: Refer to Section 41.3: WWDG implementation to check the availability of the hardware window watchdog mode.

41.4.4 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value, due to the unknown status of the prescaler when writing to the WWDG_CR register (see *Figure 859*). The *WWDG configuration register (WWDG_CFR)* contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than or equal to the window register value, and greater than 0x3F. *Figure 859* describes the window watchdog process.

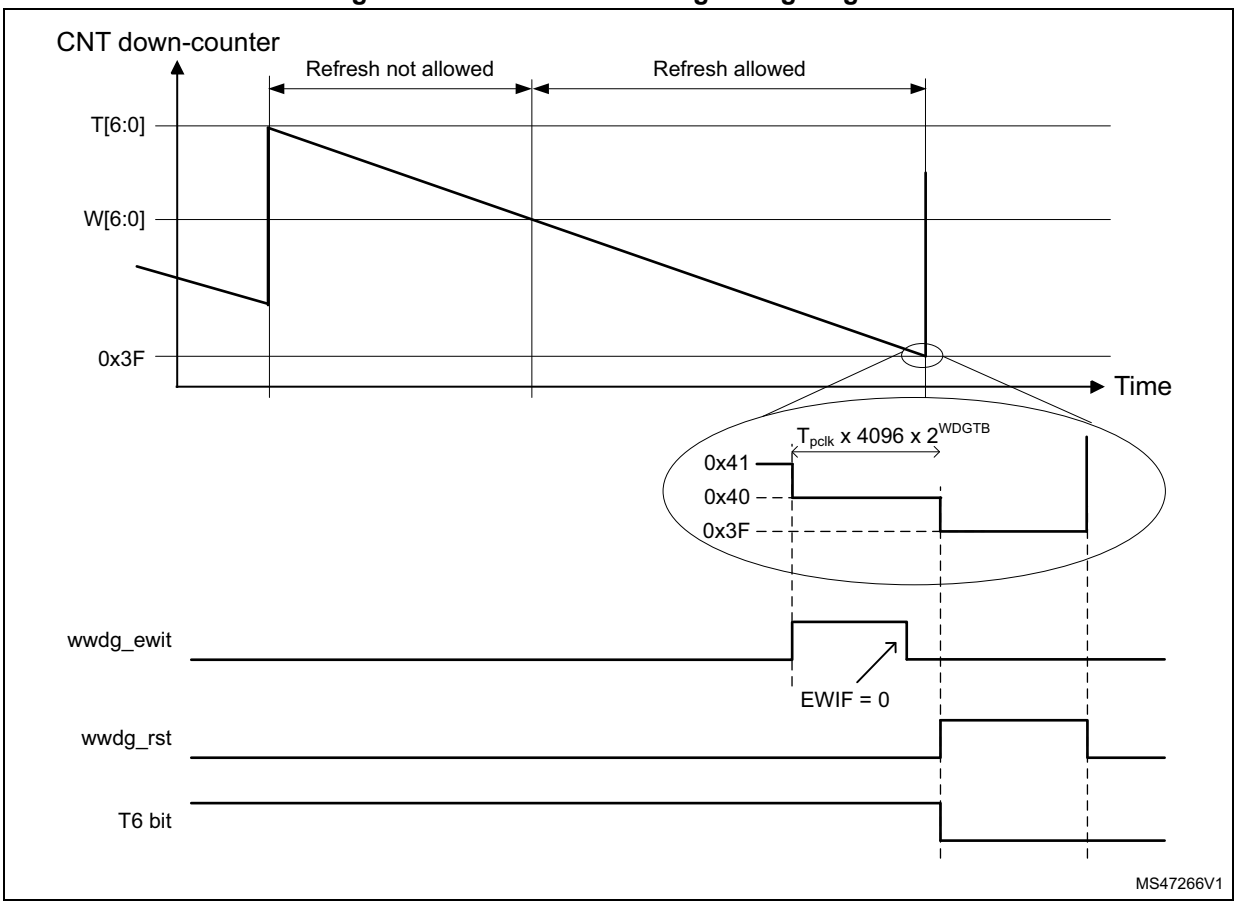
Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

41.4.5 How to program the watchdog timeout

Use the formula in *Figure 859* to calculate the WWDG timeout.

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 859. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1) \quad (ms)$$

where:

- t_{WWDG}: WWDG timeout
- t_{PCLK}: APB clock period measured in ms
- 4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[2:0] is set to 3, and T[5:0] is set to 63:

$$t_{WWDG} = (1/\ 48000) \times 4096 \times 2^{3} \times (63 + 1) = 43.69ms$$

Refer to the datasheet for the minimum and maximum values of t_{WWDG}.

41.4.6 Debug mode

When the CPU enters debug mode, WWDG counter either continues to work normally or stops, depending on debug settings. For more details refer to *Section 41.3: WWDG implementation* and to *Section 52: Debug infrastructure*.

41.5 WWDG interrupts

The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the reset is generated. To enable the early wake-up interrupt, the application must:

- Write EWIF bit of WWDG_SR register to 0, to clear unwanted pending interrupt
- Write EWI bit of WWDG_CFR register to 1, to enable interrupt

When the down-counter reaches the value 0x40, a watchdog interrupt is generated, and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case the corresponding ISR must reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The watchdog interrupt is cleared by writing 0 to the EWIF bit in the WWDG_SR register.

Note: When the watchdog interrupt cannot be served (for example due to a system lock in a higher priority task), the WWDG reset is eventually generated.

41.6 WWDG registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

41.6.1 WWDG control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

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

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

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every (4096 x $2^{WDGTB[2:0]}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

41.6.2 WWDG configuration register (WWDG_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

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

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

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:11 **WDGTB[2:0]:** Timer base

The timebase of the prescaler can be modified as follows:
000: CK counter clock (PCLK div 4096) div 1
001: CK counter clock (PCLK div 4096) div 2
010: CK counter clock (PCLK div 4096) div 4
011: CK counter clock (PCLK div 4096) div 8
100: CK counter clock (PCLK div 4096) div 16
101: CK counter clock (PCLK div 4096) div 32
110: CK counter clock (PCLK div 4096) div 64
111: CK counter clock (PCLK div 4096) div 128

Bit 10 Reserved, must be kept at reset value.

Bit 9 **EWI:** Early wake-up interrupt enable

Set by software and cleared by hardware after a reset. When set, an interrupt occurs whenever the counter reaches the value 0x40.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared with the down-counter.

41.6.3 WWDG status register (WWDG_SR)

Address offset: 0x008

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
															rc_w0

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

Bit 0 **EWIF:** Early wake-up interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing 0. Writing 1 has no effect. This bit is also set if the interrupt is not enabled.

41.6.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 477. WWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	WWDG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x004	WWDG_CFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGTB [2:0]			Res.	EWI	Res.	Res.	W[6:0]						
	Reset value																			0	0	0		0			1	1	1	1	1	1	1
0x008	WWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
	Reset value																																0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

42 Real-time clock (RTC)

42.1 RTC introduction

The RTC provides an automatic wake-up feature to manage low-power modes.

The real-time clock (RTC) is an independent counter that supports the BCD format. It provides a time-of-day clock and calendar with programmable alarm interrupts and timestamp.

The RTC is functional in V_{BAT} mode.

42.2 RTC main features

The RTC supports the following features (see *Figure 860: RTC block diagram*):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- Binary mode with 32-bit free-running counter.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- Two programmable alarms.
- On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- Timestamp feature which can be used to save the calendar content. This function can be triggered by an event on the timestamp pin, or by a tamper event, or by a switch to V_{BAT} mode.
- 17-bit auto-reload wake-up timer (WUT) for periodic events with programmable resolution and period.
- Alarm A, alarm B, wake-up Timer and timestamp individual privilege protection

The RTC is supplied through a switch that takes power either from the V_{DD} supply when present or from the VBAT pin.

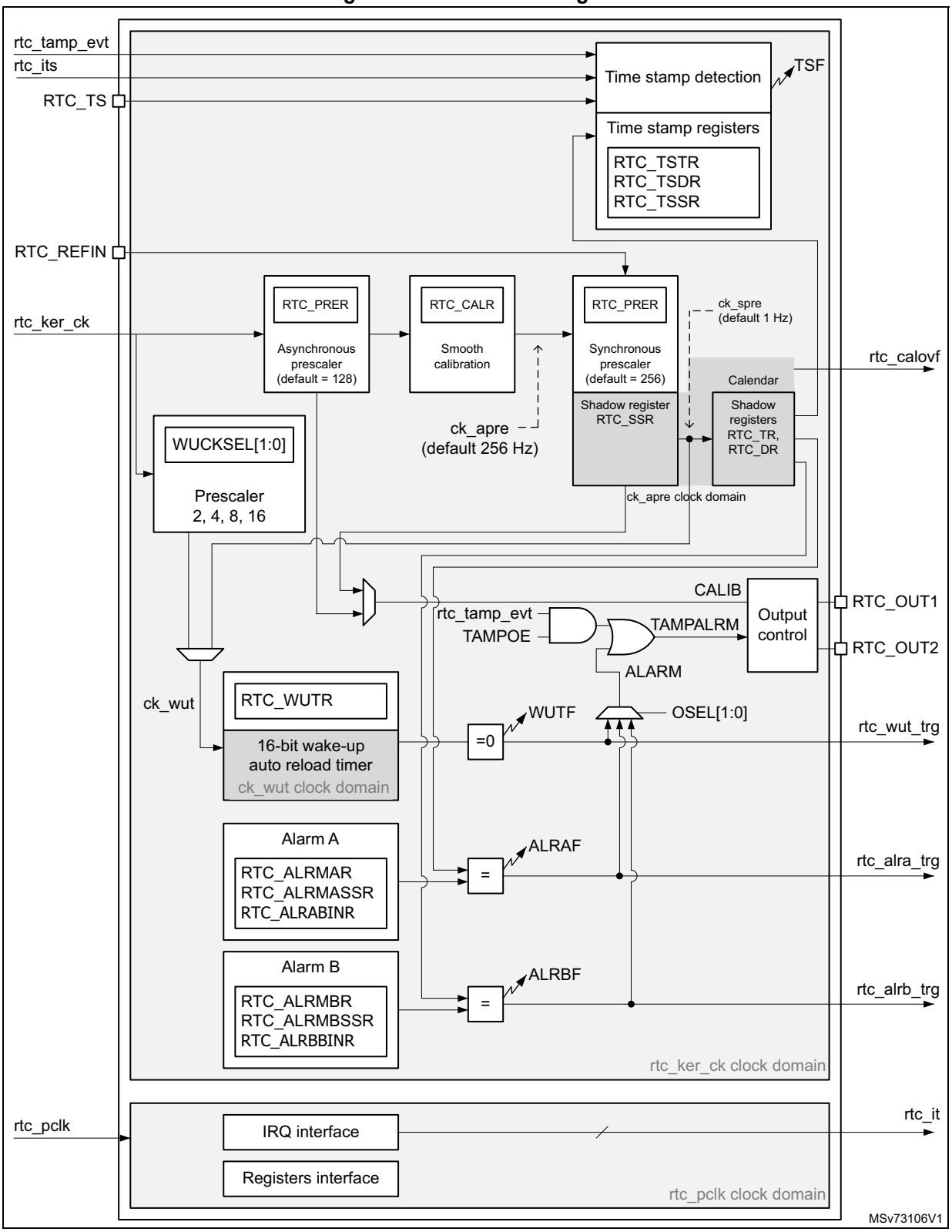
The RTC is functional in V_{BAT} mode and in the low-power modes described in the *Section 42.4: RTC low-power modes*. It is not reset by a system reset NRST.

All RTC events (Alarm, wake-up Timer, Timestamp) can generate an interrupt and wake-up the device from the supported low-power modes.

42.3 RTC functional description

42.3.1 RTC block diagram

Figure 860. RTC block diagram



42.3.2 RTC pins and internal signals

Table 478. RTC input/output pins

Pin name	Signal type	Description
RTC_TS	Input	RTC timestamp input
RTC_REFIN	Input	RTC 50 or 60 Hz reference clock input
RTC_OUT1	Output	RTC output 1
RTC_OUT2	Output	RTC output 2

RTC_OUT1 and RTC_OUT2 which selects one of the following two outputs:

- CALIB: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
- TAMPALRM: This output is the OR between rtc_tamp_evt and ALARM signals.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the alarm A, alarm B or wake-up outputs. rtc_tamp_evt is enabled by setting the TAMPOE bit in the RTC_CR register which selects the tamper event outputs.

Table 479. RTC internal input/output signals

Internal signal name	Signal type	Description
rtc_ker_ck	Input	RTC kernel clock, also named RTCCLK in this document
rtc_pclk	Input	RTC APB clock
rtc_its	Input	RTC internal timestamp event
rtc_tamp_evt	Input	Tamper event (internal or external) detected in TAMP peripheral
rtc_it	Output	RTC interrupts (refer to *Section 42.5: RTC interrupts* for details)
rtc_alra_trg	Output	RTC alarm A event detection trigger
rtc_alrb_trg	Output	RTC alarm B event detection trigger
rtc_wut_trg	Output	RTC wake-up timer event detection trigger
rtc_calovf	Output	RTC calendar overflow: this signal is generated when the RTC calendar reaches its maximum value, on the 31st of December 99, at 23:59:59. The calendar is then frozen and cannot overflow.

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes or V_{BAT} when the selected clock is not LSE. Refer to *Section 42.4: RTC low-power modes* for more details.

Table 480. RTC interconnection

Signal name	Source/destination
rtc_its	From power controller (PWR): main power loss/switch to V_{BAT} detection output
rtc_tamp_evt	From TAMP peripheral: tamp_evt
rtc_calovf	To TAMP peripheral: tamp_itamp5

The triggers outputs can be used as triggers for other peripherals.

42.3.3 GPIOs controlled by the RTC

The GPIOs with RTC functions available in Standby mode are directly controlled by the RTC, whatever the GPIO configuration.

The different RTC functions are mapped on RTC_OUT1 or on RTC_OUT2 depending on OSEL, TAMPOE, COE and OUT2EN configuration, as shown in *Table 481*.

Table 481. RTC_OUT mapping

OSEL[1:0] bits	TAMPOE	COE bit	OUT2EN bit	RTC_OUT1	RTC_OUT2
00	0	0	0	-	-
		1		CALIB	-
Don't care	1	Don't care		TAMPALRM	-
01 or 10 or 11	Don't care				-
00	0	0	1	-	-
		1		-	CALIB
Don't care	1	0		-	TAMPALRM
01 or 10 or 11	Don't care			-	
Don't care	1	1		TAMPALRM	CALIB
01 or 10 or 11	Don't care				

42.3.4 RTC registers write protection

After a system reset, the RTC registers are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set to enable RTC registers write access.

42.3.5 RTC privilege protection modes

By default after a Backup domain reset, all RTC registers can be read or written in both privileged and unprivileged modes, except for the RTC privilege mode control register (RTC_PRIVCFGR) which can be written in privilege mode only. The RTC protection configuration is not affected by a system reset.

When the PRIV bit is set in the RTC_PRIVCFGR register:

- Writing the RTC registers is possible only in privileged mode.
- Reading the RTC_PRIVCFGR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in privilege and unprivilege modes. All the other RTC registers can be read only in privilege mode.

When the PRIV bit is cleared, it is still possible to protect some of the registers by setting dedicated INITPRIV, CALPRIV, TSPRIV, WUTPRIV, ALRAPRV or ALRBPRIV control bits. If all these bits are also cleared, all the RTC registers can be read or written in privilege and unprivilege modes.

- When INITPRIV is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR and FMT control bits in RTC_CR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and unprivilege mode.
- When CALPRIV is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and unprivilege mode.
- When ALRAPRIV is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, and CALRAF and CSSRUF in the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_MISR can be read and written only in privilege mode.
- When ALRBPRIV is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE ALRBFCLR, ALRBIE in the RTC_CR, and CALRBF in the RTC_SCR, ALRBF in RTC_SR, and ALRBMF in RTC_MISR can be read and written only in privilege mode.
- When WUTPRIV is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, and CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_MISR can be read and written only in privilege mode.
- When TSPRIV is set:
  - RTC_TSTR, RTC_TSDR, RTC_TSSSR, RTC_TAMPTSCR and RTC_TSIDR registers, plus TAMPTS, ITSE, TSE, TSIE, TSEDGE control bits in the RTC_CR, CITSF, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF and ITSF in RTC_SR, and TSMF, TSOVMF and ITSMF in RTC_MISR can be read and written only in privilege mode.

An unprivileged access to a privileged-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

42.3.6 Clock and prescalers

The RTC clocks must respect this ratio: frequency(PCLK) ≥ 2 × frequency(RTCCLK).

For more information on the RTC clock (RTCCLK) source configuration, refer to “Reset and clock control (RCC)”.

BCD mode (BIN=00)

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into two programmable prescalers (see *Figure 860: RTC block diagram*):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is 2^{22}.

This corresponds to a maximum input frequency of around 4 MHz.

f_{ck_apre} is given by the following formula:

$$f_{CK_APRE} = \frac{f_{RTCCLK}}{PREDIV_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subsecond downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

f_{ck_spre} is given by the following formula:

$$f_{CK_SPRE} = \frac{f_{RTCCLK}}{(PREDIV_S + 1) \times (PREDIV_A + 1)}$$

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wake-up auto-reload timer. To obtain short timeout periods, the 16-bit wake-up auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 42.3.10: Periodic auto-wake-up* for details).

Binary mode (BIN=01)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are not functional.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.

PREDIV_S value is don’t care.

Mixed mode (BIN=10 or 11)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are also available.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register. The bits BCDU[2:0] are used to define when the calendar is incremented by 1 second, using the SSR least significant bits.

42.3.7 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ICSR register is set (see *Section 42.6.11: RTC shift control register (RTC_SHIFTR)*). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 4 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD = 0 mode, the frequency of the APB clock (f_{APB}) must be at least seven times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

42.3.8 Calendar ultra-low power mode

It is possible to reduce drastically the RTC power consumption by setting the LPCAL bit in the RTC_CALR register. In this configuration, the whole RTC is clocked by ck_apre only instead of both RTCCLK and ck_apre. Consequently, some flags delays are longer, and the calibration window is longer (refer to *Section : RTC ultra-low-power mode*).

The LPCAL bit is ignored (assumed to be 0) when asynchronous prescaler division factor (PREDIV_A+1) is not a power of 2.

Switching from LPCAL = 0 to LPCAL = 1 or from LPCAL = 1 to LPCAL = 0 is not immediate and requires a few ck_apre periods to complete.

42.3.9 Programmable alarms

The RTC unit provides programmable alarm: alarm A and alarm B. The description below is given for alarm A, but can be translated in the same way for alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register.

The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register.

When the binary mode is used, the subsecond field can be programmed in the alarm binary register RTC_ALRABINR.

The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

In case the Alarm is used to generate a trigger event for another peripheral, the ALRAF can be automatically cleared by hardware by configuring the ALRAFCLR bit at 1 in the RTC_CR register. In this configuration there is no need for software intervention if the only purpose is clearing the ALRAF flag.

Caution: If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least three to ensure correct behavior.

Alarm A and alarm B (if enabled by bits OSEL[1:0] in RTC_CR register) can be routed to the TAMPALRM output. TAMPALRM output polarity can be configured through bit POL of the RTC_CR register.

42.3.10 Periodic auto-wake-up

The periodic wake-up flag is generated by a 16-bit programmable auto-reload down-counter. The wake-up timer range can be extended to 17 bits.

The wake-up function is enabled through the WUTE bit in the RTC_CR register.

The wake-up timer clock input ck_wut can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSE (32.768 kHz), this permits the wake-up interrupt period to be configured from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock) in BCD mode, or the clock used to update the calendar as defined by BCDU in binary or mixed (BCD-binary) modes.

  When ck_spre frequency is 1 Hz, a wake-up time from 1 s to around 36 hours can be achieved with one-second resolution. This large programmable time range is divided in two parts:
  - from 1 s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18 h to 36 h when WUCKSEL[2:1] = 11. In this last case 2^{16} is added to the 16-bit counter current value. When the initialization sequence is complete (see *Programming the wake-up timer on page 2457*), the timer starts counting down. When the wake-up function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_SR register, and the wake-up counter is automatically reloaded with its reload value (RTC_WUTR register value).

Depending on WUTOCLR in the RTC_WUTR register, the WUTF flag must either be cleared by software (WUTOCLR = 0x0000), or the WUTF is automatically cleared by hardware when the auto-reload down counter reaches WUTOCLR value (0x0000 < WUTOCLR ≤ WUT).

The wake-up flag is output on an internal signal rtc_wut that can be used by other peripherals (refer to section *Section 42.3.1: RTC block diagram*).

When the periodic wake-up interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wake-up flag can be routed to the TAMPALRM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. TAMPALRM output polarity can be configured through the POL bit in the RTC_CR register.

42.3.11 RTC initialization and configuration

RTC Binary, BCD or Mixed mode

By default the RTC is in BCD mode (BIN = 00 in the RTC_ICSR register): the RTC_SSR register contains the subsecond field SS[15:0], clocked by ck_apre, allowing to generate a 1 Hz clock to update the calendar registers in BCD format (RTC_TR and RTC_DR).

When the RTC is configured in binary mode (BIN = 01 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar registers in BCD format (RTC_TR and RTC_DR) are not used.

When the RTC is configured in mixed mode (BIN = 10 or 11 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar is updated (1 second increment) each time the SSR[BCDU+7:0] reaches 0.

RTC write protection key

After Backup domain reset, the RTC registers related to RTC initialization and RTC calibration are write-protected: Writing the unlock sequence in RTC_WPR is required to gain write access to these registers.

The following steps are required to unlock the write protection on the registers related to RTC initialization that is RTC_TR, RTC_DR, RTC_PRER, the bits INIT, BIN and BCDU in RTC_ICSR, and the bit FMT in RTC_CR:

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Those two write accesses must be performed with the privilege mode defined by PRIV, INITPRIV configuration

Writing an incorrect key reactivates the write protection.

Any write access to RTC_WPR with incorrect privilege settings has no effect and is ignored without any illegal access notification.

The following steps are required to unlock the write protection of the registers related to RTC calibration, that is RTC_CALR, RTC_SHIFTR, and the bits SUB1H, ADD1H, REFCKON in RTC_CR:

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Those two write accesses must be done with the privilege mode defined by PRIV, CALPRIV configuration.

Writing a wrong key reactivates the write protection.

Any write access to RTC_WPR with the wrong privilege settings has no effect and is ignored without any illegal access notification.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ICSR register. The initialization phase mode is entered when INITF is set to 1.
   – If LPCAL = 0: INITF is set around two RTCCLK cycles after INIT bit is set.
   – If LPCAL = 1: INITF is set up to two ck_apre cycle after INIT bit is set.
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register, plus BIN and BCDU in the RTC_ICSR register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded.
   – If LPCAL = 0: the counting restarts after four RTCCLK clock cycles.
   – If LPCAL = 1: the counting restarts after up to two RTCCLK + 1 ck_apre.

When the initialization sequence is complete, the calendar starts counting. The RTC_SSR content is initialized with:

- PREDIV_S in BCD mode (BIN = 00)
- 0xFFFF FFFF in binary or mixed (BCD-binary) modes (BIN = 01, 10 or 11).

In BCD mode, RTC_SSR contains the value of the synchronous prescaler counter. This enables one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]). The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption. The RTC dynamic consumption is optimized for PREDIV_A+1 being a power of 2.

Note: There is no hardware protection against incorrect calendar value initialization. Software must ensure that the initialization is performed correctly using allowed calendar BCD values.

Note: After a system reset, the application can read the INITS flag in the RTC_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its Backup domain reset default value (0x00).

Note: To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A but can be translated in the same way for alarm B.

1. Clear ALRAE in RTC_CR to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASSR/RTC_ALRMAR or RTC_ALRABINR).
3. Set ALRAE in the RTC_CR register to enable alarm A again.

Note: Each change of the RTC_CR register is taken into account after around two RTCCLK clock cycles due to clock synchronization.

Programming the wake-up timer

The following sequence is required to configure or change the wake-up timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wake-up timer.
2. Poll WUTWF until it is set in RTC_ICSR to make sure the access to wake-up auto-reload counter and to WUCKSEL[2:0] bits is allowed.
   - If WUCKSEL[2] = 0: WUTWF is set around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is cleared.
   - If WUCKSEL[2] = 1: WUTWF is set up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is cleared.
3. Program the wake-up auto-reload value WUT[15:0], WUTOCLR[15:0] and the wake-up clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wake-up timer restarts down-counting.
   - If WUCKSEL[2] = 0: WUTWF is cleared around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is set.
   - If WUCKSEL[2] = 1: WUTWF is cleared up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is set.

42.3.12 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB clock frequency (f_{PCLK}) must be equal to or greater than seven times the RTC clock frequency (f_{RTCCLK}). This ensures a secure behavior of the synchronization mechanism.

If the APB clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB clock frequency must never be lower than two times the RTC clock frequency.

The RSF bit is set in RTC_ICSR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycle. To ensure consistency between the three values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller

than one RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wake-up and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to *Calendar initialization and configuration on page 2456*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 42.3.14: RTC synchronization*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (Stop or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the two readings. If the values differ, repeat the process until matching results are obtained. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD = 1, instructions which read the calendar registers require one extra APB cycle to complete.

42.3.13 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ICSR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a Backup domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the wake-up timer register (RTC_WUTR), the alarm A and alarm B registers (RTC_ALRMASSR/RTC_ALRMAR/RTC_ALRABINR and RTC_ALRMBSSR/RTC_ALRMBR/RTC_ALRBBINR).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one (refer to RCC for details about RTC

clock sources not affected by system reset). When a Backup domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

42.3.14 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the subsecond field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock with a resolution of one ck_apre period.

The shift operation consists in adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock.

If at the same time the ADD1S bit is set in BCD or mixed mode, this results in adding one second and at the same time subtracting a fraction of second, so this advances the clock. ADD1S has no effect in binary mode.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

Caution: In mixed mode (BIN = 10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Caution: Before initiating a shift operation in BCD mode, the user must check that SS[15] = 0 in order to ensure that no overflow occurs. In mixed mode, the user must check that the bit SS[BCDU+8] = 0.

Caution: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

42.3.15 RTC reference clock detection

This feature is available only in BCD mode (BIN = 00).

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock is higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals seven ck_apre periods when detecting the first reference clock edge. A smaller window of three ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREVID_S = 0x00FF

42.3.16 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual ck_cal pulses).

If LPCAL = 0: ck_cal = RTCCLK

If LPCAL = 1: ck_cal = ck_apre

These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

RTC ultra-low-power mode

The RTC consumption can be reduced by setting the LPCAL bit in the RTC calibration register (RTC_CALR). In this case, the calibration mechanism is applied on ck_apre instead of RTCCLK. The resulting accuracy is the same, but the calibration is performed during a calibration cycle of about 2^{20} x PREDIV_A x RTCCLK pulses instead of 2^{20} RTCCLK pulses when LPCAL = 0.

Smooth calibration mechanism

The smooth calibration register (RTC_CALR) specifies the number of ck_cal clock cycles to be masked during the calibration cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

Note: CALM[8:0] (RTC_CALR) specifies the number of ck_cal pulses to be masked during the calibration cycle. Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1] = 1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2] = 1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8] = 1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM permits the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to 1 effectively inserts an extra ck_cal pulse every 2^{11} ck_cal cycles, which means that 512 clocks are added during every calibration cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 ck_cal cycles can be added during the calibration cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Caution: PREDIV_A must be greater or equal to 3.

Calibration when PREDIV_A < 3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

It is however possible to perform a calibration with PREDIV_A less than 3 in BCD mode, the synchronous prescaler value (PREDIV_S) is reduced so that each second is accelerated by 8 ck_cal clock cycles, which is equivalent to adding 256 clock cycles every calibration cycle. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each calibration cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S is set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S is set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Verifying the RTC calibration

It is recommended to verify the RTC calibration with LPCAL = 0, in order to have a 32-second calibration cycle.

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to two RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

  Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).
- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16-second calibration cycle period.

  In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.
- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.

  In this case, the RTC precision can be measured during eight seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Recalibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

42.3.17 Timestamp function

Timestamp is enabled by setting the TSE or ITSE bits of RTC_CR register to 1.

When TSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the RTC_TS pin.

When TAMPTS or any bit of RTC_TAMPTSCR is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal or external tamper event is detected. Refer to *RTC control register (RTC_CR)* and *RTC timestamp on tamper control register (RTC_TAMPTSCR)*.

When ITSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal timestamp event is detected. The internal timestamp event is generated by the switch to the V_{BAT} supply.

When a timestamp event occurs, due to internal or external event, the timestamp flag bit (TSF) in RTC_SR register is set. In case the event is internal, the ITSF flag is also set in RTC_SR register. The TSID bitfield in the RTC_TSIDR provides the identification of the cause of the timestamp event. It is mandatory to check ITSF after clearing TSF status flag,

in case an internal event occurs after first timestamp event. If set, ITSF must also be cleared by software to allow next timestamp event correct identification.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

Note: TSF is set up to two ck_apre cycles after the timestamp event from RTC_TS pin or from rtc_its internal signal occurs, due to synchronization process. TSF is set up to three ck_apre cycles after tamper flags. TSID indicates the source corresponding to time stored in timestamp registers, but due to this difference of timing and priority between events, RTC_TS or rtc_its can be identified as first event, even if it occurs up to one ck_apre after TAMP event.

TSOVF is set up to only one ck_apre cycle after the event occurs. This means that if two timestamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

Caution: If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write 0 into TSF bit unless it has already read it to 1.

42.3.18 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the CALIB frequency is $f_{RTCCLK}/(256 * (PREDIV_A+1))$. This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = Ox7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

Note: When COSEL is cleared, the CALIB output is the output of the 6^{th} stage of the asynchronous prescaler. If LPCAL is changed from 0 to 1, the output can be irregular (glitch...) during the LPCAL switch. If LPCAL = 1 this output is always available. If LPCAL = 0, no output is present if PREDIV_A is < 0x20.

When COSEL is set, the CALIB output is the output of the 8^{th} stage of the synchronous prescaler.

42.3.19 Tamper and alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_SR register.

When the TAMPOE control bit is set in the RTC_CR, all external and internal tamper flags are ORed and routed to the TAMPALRM output. If OSEL = 00 the TAMPALRM output reflects only the tampers flags. If OSEL ≠ 00, the signal on TAMPALRM provides both tamper flags and alarm A, B, or wake-up flag.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC_CR so that the opposite of the selected flags bit is output when POL is set to 1.

TAMPALRM output

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM_TYPE in the RTC_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM_PU in the RTC_CR.

Note: Once the TAMPALRM *output is enabled, it has priority over CALIB on RTC_OUT1.*

42.4 RTC low-power modes

Table 482. Effect of low-power modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Stop mode.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Standby mode.

The table below summarizes the RTC pins and functions capability in all modes.

Table 483. RTC pins functionality over modes

Functions	Functional in all low-power modes except Standby mode	Functional in Standby mode	Functional in V_{BAT} mode
RTC_TS	Yes	Yes	Yes
RTC_REFIN	Yes	No	No
RTC_OUT1	Yes	Yes	Yes
RTC_OUT2	Yes	No	No

42.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register. The interrupt output is also activated.

The RTC provides an interrupt to a specific CPU only if its CID filter is disabled, or if it is enabled and configured with the CPU CID in the RTC_RxCIDCFGR register (x = 0 to 3).

Table 484. Interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from low-power modes
RTC	Alarm A	ALRAF	ALRAIE	Write 1 in CALRAF	Yes(3)
	Alarm B	ALRBF	ALRBIE	Write 1 in CALRBF	Yes(3)
	Timestamp	TSF	TSIE	Write 1 in CTSF	Yes(3)
	Wake-up timer	WUTF	WUTIE	Write 1 in CWUTF	Yes(3)
	SSR underflow (reload)	SSRUF	SSRUIE	Write 1 in CSSRUF	Yes(3)

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_MISR register.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

42.6 RTC registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

42.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 2456* and *Reading the calendar on page 2457*.

This register is write protected. The write access procedure is described in *RTC write protection key on page 2455*.

This register can be write-protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

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

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

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

42.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 2456* and *Reading the calendar on page 2457*.

This register is write protected. The write access procedure is described in *RTC write protection key on page 2455*.

This register can be write-protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset value: 0x0000 2101 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units
000: forbidden
001: Monday
...
111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

Note: The calendar is frozen when reaching the maximum value, and can't roll over.

42.6.3 RTC subsecond register (RTC_SSR)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

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

Bits 31:0 **SS[31:0]**: Synchronous binary counter

SS[31:16]: Synchronous binary counter MSB values
When Binary or Mixed mode is selected (BIN = 01 or 10 or 11):
SS[31:16] are the 16 MSB of the SS[31:0] free-running down-counter.
When BCD mode is selected (BIN = 00):
SS[31:16] are forced by hardware to 0x0000.
SS[15:0]: Subsecond value/synchronous binary counter LSB values
When Binary mode is selected (BIN = 01 or 10 or 11):
SS[15:0] are the 16 LSB of the SS[31:0] free-running down-counter.
When BCD mode is selected (BIN = 00):
SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:
Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)
SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

42.6.4 RTC initialization control and status register (RTC_ICSR)

This register is write protected. The write access procedure is described in *RTC write protection key on page 2455*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x0C

Backup domain reset value: 0x0000 0007

System reset: not affected except INIT, INITF, and RSF bits which are cleared to 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BCDU[2:0]			BIN[1:0]		INIT	INITF	RSF	INITS	SHPF	WUTWF	Res.	Res.
			rw	rw	rw	rw	rw	rw	r	rc_w0	r	r	r		

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to *Recalibration on-the-fly*.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:10 **BCDU[2:0]**: BCD update (BIN = 10 or 11)

In mixed mode when both BCD calendar and binary extended counter are used (BIN = 10 or 11), the calendar second is incremented using the SSR Least Significant Bits.

0x0: 1s calendar increment is generated each time SS[7:0] = 0
0x1: 1s calendar increment is generated each time SS[8:0] = 0
0x2: 1s calendar increment is generated each time SS[9:0] = 0
0x3: 1s calendar increment is generated each time SS[10:0] = 0
0x4: 1s calendar increment is generated each time SS[11:0] = 0
0x5: 1s calendar increment is generated each time SS[12:0] = 0
0x6: 1s calendar increment is generated each time SS[13:0] = 0
0x7: 1s calendar increment is generated each time SS[14:0] = 0

Bits 9:8 **BIN[1:0]**: Binary mode

00: Free running BCD calendar mode (Binary mode disabled).
01: Free running Binary mode (BCD mode disabled)
10: Free running BCD calendar and Binary modes
11: Free running BCD calendar and Binary modes

Bit 7 **INIT**: Initialization mode

0: Free running mode
1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER), plus BIN and BCDU fields. Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSR, RTC_TR and RTC_DR). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPSHAD = 1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Note: RSF is not set when the frequency of the APB clock is less than twice the frequency of RTCCLK.

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

0: No shift operation is pending

1: A shift operation is pending

Bit 2 **WUTWF**: Wake-up timer write flag

This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC_CR.

0: Wake-up timer configuration update not allowed

1: Wake-up timer configuration update allowed

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

42.6.5 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to *Calendar initialization and configuration on page 2456*.

This register is write protected. The write access procedure is described in *RTC write protection key on page 2455*.

This register can be write-protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x10

Backup domain reset value: 0x007F 00FF

System reset: not affected

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

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

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor

This is the asynchronous division factor:
ck_apre frequency = RTCCLK frequency/(PREDIV_A+1)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor

This is the synchronous division factor:
ck_spre frequency = ck_apre frequency/(PREDIV_S+1)

42.6.6 RTC wake-up timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ICSR.

This register can be protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x14

Backup domain reset value: 0x0000 FFFF

System reset: not affected

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

Bits 31:16 **WUTOCLR[15:0]**: Wake-up auto-reload output clear value

When WUTOCLR[15:0] is different from 0x0000, WUTF is set by hardware when the auto-reload down-counter reaches 0 and is cleared by hardware when the auto-reload downcounter reaches WUTOCLR[15:0].

When WUTOCLR[15:0] = 0x0000, WUTF is set by hardware when the WUT down-counter reaches 0 and is cleared by software.

Bits 15:0 **WUT[15:0]**: Wake-up auto-reload value bits

When the wake-up timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register.

When WUCKSEL[2] = 1, the wake-up timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs between WUT and (WUT + 2) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

42.6.7 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in RTC write protection key on page 2455.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OUT2EN**: RTC_OUT2 output enable

With this bit set, the RTC outputs can be remapped on RTC_OUT2 as follows:
OUT2EN = 0: RTC output 2 disable
If OSEL ≠ 00 or TAMPOE = 1: TAMPALRM is output on RTC_OUT1
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT1
OUT2EN = 1: RTC output 2 enable
If (OSEL ≠ 00 or TAMPOE = 1) and COE = 0: TAMPALRM is output on RTC_OUT2
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT2
If (OSEL ≠ 00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC_OUT2 and TAMPALRM is output on RTC_OUT1.

Bit 30 **TAMPALRM_TYPE**: TAMPALRM output type

0: TAMPALRM is push-pull output
1: TAMPALRM is open-drain output

Bit 29 **TAMPALRM_PU**: TAMPALRM pull-up enable

0: No pull-up is applied on TAMPALRM output
1: A pull-up is applied on TAMPALRM output

Bit 28 **ALRBFCLR**: Alarm B flag automatic clear

0: Alarm B event generates a trigger event and ALRBF must be cleared by software to allow next alarm event.
1: Alarm B event generates a trigger event. ALRBF is automatically cleared by hardware after one ck_apre cycle.

Bit 27 **ALRAFCLR**: Alarm A flag automatic clear

0: Alarm A event generates a trigger event and ALRAF must be cleared by software to allow next alarm event.
1: Alarm A event generates a trigger event. ALRAF is automatically cleared by hardware after one ck_apre cycle.

Bit 26 **TAMPOE**: Tamper detection output enable on TAMPALRM

0: The tamper flag is not routed on TAMPALRM
1: The tamper flag is routed on TAMPALRM, combined with the signal provided by OSEL and with the polarity provided by POL.

Bit 25 **TAMPTS**: Activate timestamp on tamper detection event

0: Tamper detection event does not cause a RTC timestamp to be saved if none of the RTC_TAMPTSCR bit is set
1: Save RTC timestamp on tamper detection event (for any tamper event, whatever RTC_TAMPTSCR value)

TAMPTS is valid even if TSE = 0 in the RTC_CR register. Timestamp flag is set up to three ck_apre cycles after the tamper flags.

Note: TAMPTS must be cleared before entering RTC initialization mode.

Bit 24 **ITSE**: timestamp on internal event enable

0: internal event timestamp disabled
1: internal event timestamp enabled

Bit 23 **COE**: Calibration output enable

This bit enables the CALIB output
0: Calibration output disabled
1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection
These bits are used to select the flag to be routed to TAMPALRM output.
00: Output disabled
01: Alarm A output enabled
10: Alarm B output enabled
11: Wake-up output enabled

Bit 20 **POL**: Output polarity
This bit is used to configure the polarity of TAMPALRM output.
0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).
1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

Bit 19 **COSEL**: Calibration output selection
When COE = 1, this bit selects which signal is output on CALIB.
0: Calibration output is 512 Hz
1: Calibration output is 1 Hz
These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A = 127 and PREDIV_S = 255). Refer to *Section 42.3.18: Calibration clock output*.

Bit 18 **BKP**: Backup
This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: *S*ubtract 1 hour (winter time change)
When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.
Setting this bit has no effect when current hour is 0.
0: No effect
1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)
When this bit is set outside initialization mode, one hour is added to the calendar time. This bit is always read as 0.
0: No effect
1: Adds 1 hour to the current time.
Note: This can be used for summer time change
ADD1H is ignored when SUB1H is set.

Bit 15 **TSIE**: Timestamp interrupt enable
0: Timestamp interrupt disable
1: Timestamp interrupt enable

Bit 14 **WUTIE**: Wake-up timer interrupt enable
0: Wake-up timer interrupt disabled
1: Wake-up timer interrupt enabled

Bit 13 **ALRBIE**: Alarm B interrupt enable
0: Alarm B interrupt disable
1: Alarm B interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable
0: Alarm A interrupt disabled
1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable
0: Timestamp disable
1: Timestamp enable

Bit 10 **WUTE**: Wake-up timer enable
0: Wake-up timer disabled
1: Wake-up timer enabled
Note: When the wake-up timer is disabled, wait for WUTWF = 1 before enabling it again.

Bit 9 **ALRBE**: Alarm B enable
0: Alarm B disabled
1: Alarm B enabled

Bit 8 **ALRAE**: Alarm A enable
0: Alarm A disabled
1: Alarm A enabled

Bit 7 **SSRUIE**: SSR underflow interrupt enable
0: SSR underflow interrupt disabled
1: SSR underflow interrupt enabled

Bit 6 **FMT**: Hour format
0: 24 hour/day format
1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers
0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.
1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.
Note: If the frequency of the APB clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to 1.

Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)
0: RTC_REFIN detection disabled
1: RTC_REFIN detection enabled
Note: BIN must be 0x00 and PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Timestamp event active edge
0: RTC_TS input rising edge generates a timestamp event
1: RTC_TS input falling edge generates a timestamp event
TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: ck_wut wake-up clock selection
000: RTC/16 clock is selected
001: RTC/8 clock is selected
010: RTC/4 clock is selected
011: RTC/2 clock is selected
10x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU.
11x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU. Furthermore, 2^{16} is added to the WUT counter value.

Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC_ICSR/INITF = 1).

WUT = wake-up unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ICSR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it may mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

42.6.8 RTC privilege mode control register (RTC_PRIVCFGR)

This register can be written only when the APB access is privileged.

Address offset: 0x1C

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PRIV**: RTC privilege protection

0: All RTC registers can be written when the APB access is privileged or unprivileged, except the registers protected by other privilege protection bits.

1: All RTC registers can be written only when the APB access is privileged.

Bit 14 **INITPRIV**: Initialization privilege protection

0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is privileged or unprivileged.

1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is privileged.

Bit 13 **CALPRIV**: Shift register, daylight saving, calibration and reference clock privilege protection

0: Shift register, daylight saving, calibration and reference clock can be written when the APB access is privileged or unprivileged.

1: Shift register, daylight saving, calibration and reference clock can be written only when the APB access is privileged.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSPRIV**: Timestamp privilege protection

0: RTC Timestamp configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Timestamp configuration and interrupt clear can be written only when the APB access is privileged.

Bit 2 **WUTPRIV**: Wake-up timer privilege protection

0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC wake-up timer configuration and interrupt clear can be written only when the APB access is privileged.

Bit 1 **ALRBPRIV**: Alarm B privilege protection

0: RTC Alarm B configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Alarm B configuration and interrupt clear can be written only when the APB access is privileged.

Bit 0 **ALRAPRIV**: Alarm A and SSR underflow privilege protection

0: RTC Alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is privileged.

Note: Refer to Section 42.3.5: RTC privilege protection modes for details on the read protection.

This register can be globally write-protected, or each bit of this register can be individually write-protected against unprivileged access depending on the RTC_PRIVCFGR configuration (refer to *Section 42.3.5: RTC privilege protection modes*).

42.6.9 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

Bits 7:0 **KEY[7:0]**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to *RTC write protection key* for a description of how to unlock RTC register write protection.

42.6.10 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC write protection key on page 2455*.

This register can be write-protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x28

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW 16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm
0: No RTCCLK pulses are added.
1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm).
This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows:
(512 × CALP) - CALM.
Refer to *Section 42.3.16: RTC smooth digital calibration*.

Bit 14 **CALW8**: Use an 8-second calibration cycle period
When CALW8 is set to 1, the 8-second calibration cycle period is selected.
Note: CALM[1:0] are stuck at 00 when CALW8 = 1. Refer to Section 42.3.16: RTC smooth digital calibration.

Bit 13 **CALW16**: Use a 16-second calibration cycle period
When CALW16 is set to 1, the 16-second calibration cycle period is selected. This bit must not be set to 1 if CALW8 = 1.
Note: CALM[0] is stuck at 0 when CALW16 = 1. Refer to Section 42.3.16: RTC smooth digital calibration.

Bit 12 **LPCAL**: RTC low-power mode

0: Calibration window is 2^{20} RTCCLK, which is a high-consumption mode. This mode must be set only when less than 32s calibration window is required.

1: Calibration window is 2^{20} ck_apre, which is the required configuration for ultra-low consumption mode.

Bits 11:9 Reserved, must be kept at reset value.

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature must be used in conjunction with CALP. See *Section 42.3.16: RTC smooth digital calibration on page 2460*.

42.6.11 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC write protection key on page 2455*.

This register can be protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															

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

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value.

Bits 14:0 **SUBFS[14:0]**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

Advance (seconds) = 1 - (SUBFS / (PREDIV_S + 1)).

In mixed BCD-binary mode (BIN = 10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF = 1 to be sure that the shadow registers have been updated with the shifted time.

42.6.12 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

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

Bit 22 **PM**: AM/PM notation

0: AM or 24-hour format

1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

42.6.13 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **WDU[2:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

42.6.14 RTC timestamp subsecond register (RTC_TSSSR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when the TSF bit is reset.

This register can be protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x38

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 SS[31:0]: Subsecond value/synchronous binary counter values

SS[31:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

42.6.15 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAE is reset in RTC_CR register.

This register can be protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x40

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm A date mask

0: Alarm A set if the date/day match

1: Date/day don't care in alarm A comparison

Bit 30 **WDSEL**: Week day selection

0: DU[3:0] represents the date units

1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm A hours mask
0: Alarm A set if the hours match
1: Hours don't care in alarm A comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm A minutes mask
0: Alarm A set if the minutes match
1: Minutes don't care in alarm A comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm A seconds mask
0: Alarm A set if the seconds match
1: Seconds don't care in alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

42.6.16 RTC alarm A subsecond register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register.

This register can be protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x44

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)

0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.

1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRABINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRABINR.SS[31:0].

Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit

0: No comparison on subseconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[31:1] are don't care in Alarm A comparison. Only SS[0] is compared.

2: SS[31:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.

...

31: SS[31] is don't care in Alarm A comparison. Only SS[30:0] are compared.

From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

Note: In BCD mode (BIN = 00) the overflow bits of the synchronous counter (bits 31:15) are never compared. These bits can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

This field is the mirror of SS[14:0] in the RTC_ALRABINR, and so can also be read or written through RTC_ALRABINR.

42.6.17 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBE is reset in RTC_CR register.

This register can be protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x48

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WD SEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm B date mask
0: Alarm B set if the date and day match
1: Date and day don't care in alarm B comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm B hours mask
0: Alarm B set if the hours match
1: Hours don't care in alarm B comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm B minutes mask
0: Alarm B set if the minutes match
1: Minutes don't care in alarm B comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm B seconds mask
0: Alarm B set if the seconds match
1: Seconds don't care in alarm B comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

42.6.18 RTC alarm B subsecond register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register.

This register can be protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x4C

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)

0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.

1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRBBINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRBBINR.SS[31:0].

Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit

0: No comparison on subseconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[31:1] are don't care in Alarm B comparison. Only SS[0] is compared.

2: SS[31:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.

...

31: SS[31] is don't care in Alarm B comparison. Only SS[30:0] are compared.

From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

Note: In BCD mode (BIN = 00)The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

This field is the mirror of SS[14:0] in the RTC_ALRBBINR, and so can also be read or written through RTC_ALRBBINR.

42.6.19 RTC status register (RTC_SR)

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x50

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUF**: SSR underflow flag

This flag is set by hardware when the SSR is reloaded with 0xFFFF FFFF after reaching 0. SSRUF is not set when SSCLR = 1.

Note: SSRUF is not an error event as SSR counter is a free-running down-counter with automatic reload. SSRUF is set only in binary and mixed modes. It is not set in BCD mode.

Bit 5 **ITSF**: Internal timestamp flag

This flag is set by hardware when a timestamp on the internal event occurs.

Bit 4 **TSOVF**: Timestamp overflow flag

This flag is set by hardware when a timestamp event occurs while TSF is already set.

It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSF**: Timestamp flag

This flag is set by hardware when a timestamp event occurs.

If ITSF flag is set, TSF must be cleared together with ITSF.

Note: TSF is not set if TAMPTS = 1 and the tamper flag is read during the 3 ck_apre cycles following tamper event.

Bit 2 **WUTF**: Wake-up timer flag

This flag is set by hardware when the wake-up auto-reload counter reaches 0.

If WUTOCLR[15:0] is different from 0x0000, WUTF is cleared by hardware when the wake-up auto-reload counter reaches WUTOCLR value.

If WUTOCLR[15:0] is 0x0000, WUTF must be cleared by software.

This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBF**: Alarm B flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm B register (RTC_ALRMBR).

Bit 0 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm A register (RTC_ALRMAR).

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

42.6.20 RTC masked interrupt status register (RTC_MISR)

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x54

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow masked flag
This flag is set by hardware when the SSR underflow interrupt occurs.

Bit 5 **ITSMF**: Internal timestamp masked flag
This flag is set by hardware when a timestamp on the internal event occurs and timestampinterrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow masked flag
This flag is set by hardware when a timestamp interrupt occurs while TSMF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp masked flag
This flag is set by hardware when a timestamp interrupt occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wake-up timer masked flag
This flag is set by hardware when the wake-up timer interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B masked flag
This flag is set by hardware when the alarm B interrupt occurs.

Bit 0 **ALRAMF**: Alarm A masked flag
This flag is set by hardware when the alarm A interrupt occurs.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

42.6.21 RTC status clear register (RTC_SCR)

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x5C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSSRUF**: Clear SSR underflow flag
Writing ‘1’ in this bit clears the SSRUF in the RTC_SR register.

Bit 5 **CITSF**: Clear internal timestamp flag
Writing 1 in this bit clears the ITSF bit in the RTC_SR register.

Bit 4 **CTSOVF**: Clear timestamp overflow flag
Writing 1 in this bit clears the TSOVF bit in the RTC_SR register.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **CTSF**: Clear timestamp flag
Writing 1 in this bit clears the TSF bit in the RTC_SR register.
If ITSF flag is set, TSF must be cleared together with ITSF by setting CRSF and CITSF.

Bit 2 **CWUTF**: Clear wake-up timer flag
Writing 1 in this bit clears the WUTF bit in the RTC_SR register.

Bit 1 **CALRBF**: Clear alarm B flag
Writing 1 in this bit clears the ALRBF bit in the RTC_SR register.

Bit 0 **CALRAF**: Clear alarm A flag
Writing 1 in this bit clears the ALRAF bit in the RTC_SR register.

42.6.22 RTC timestamp on tamper control register (RTC_TAMPTSCR)

Address offset: 0x64

Backup domain reset value: 0x0000 0000

System reset: not affected

This register can be protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5TS	TAMP4TS	TAMP3TS	TAMP2TS	TAMP1TS
											rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **ITAMPTS**: Timestamp on internal tamper event
- 0: When TAMPTS = 0, internal tamper detection event does not cause a RTC timestamp to be saved
- 1: Save RTC timestamp on any internal tamper detection event (whatever TAMPTS value)

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TAMP5TS**: Timestamp on external tamper TAMP5 event
- 0: When TAMPTS = 0, TAMP5 detection event does not cause an RTC timestamp to be saved
- 1: Save RTC timestamp on TAMP5 detection event (whatever TAMPTS value)

Bit 3 **TAMP4TS**: Timestamp on external tamper TAMP4 event
- 0: When TAMPTS = 0, TAMP4 detection event does not cause an RTC timestamp to be saved
- 1: Save RTC timestamp on TAMP4 detection event (whatever TAMPTS value)

Bit 2 **TAMP3TS**: Timestamp on external tamper TAMP3 event
- 0: When TAMPTS = 0, TAMP3 detection event does not cause an RTC timestamp to be saved
- 1: Save RTC timestamp on TAMP3 detection event (whatever TAMPTS value)

Bit 1 **TAMP2TS**: Timestamp on external tamper TAMP2 event
- 0: When TAMPTS = 0, TAMP2 detection event does not cause an RTC timestamp to be saved
- 1: Save RTC timestamp on TAMP2 detection event (whatever TAMPTS value)

Bit 0 **TAMP1TS**: Timestamp on external tamper TAMP1 event
- 0: When TAMPTS = 0, TAMP1 detection event does not cause an RTC timestamp to be saved
- 1: Save RTC timestamp on TAMP1 detection event (whatever TAMPTS value)

Note: All bits of RTC_TAMPTSCR must be cleared before entering RTC initialization mode.

42.6.23 RTC timestamp status register (RTC_TSIDR)

The content of this register is valid only when TSF is set to 1 in RTC_SR.

Address offset: 0x68

Backup domain reset value: 0x0000 0000

System reset: not affected

This register can be protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

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

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

Bits 5:0 **TSID[5:0]**: Timestamp flag source identification

This bitfield is updated with identification of the source generating the timestamp stored in timestamp registers. It is updated at each TSF event, and is not affected by TSOVF event.

000000: RTC_TS pin event
000001: TAMP1F event
000010: TAMP2F event
000011: TAMP3F event
000100: TAMP4F event
000101: TAMP5F event
010000: Internal tamper event
100000: Internal timestamp event (from rtc_its internal signal)
Others: Reserved

42.6.24 RTC alarm A binary mode register (RTC_ALRABINR)

This register can be written only when ALRAE is reset in RTC_CR register.

This register can be protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x70

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value

This value is compared with the contents of the synchronous counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.
SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMASSR, and so can also be read or written through RTC_ALRMASSR.

42.6.25 RTC alarm B binary mode register (RTC_ALRBBINR)

This register can be written only when ALRBE is reset in RTC_CR register.

This register can be protected against unprivileged access. Refer to *Section 42.3.5: RTC privilege protection modes*.

Address offset: 0x74

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value

This value is compared with the contents of the synchronous counter to determine if Alarm Bis to be activated. Only bits 0 up MASKSS-1 are compared.
SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMBSSR, and so can also be read or written through RTC_ALRMBSSR.

42.6.26 RTC register map

Table 485. RTC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RTC_TR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	RTC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1			0	0	0	0	0	1
0x08	RTC_SSR	SS[31:16]																SS[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	RTC_ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF	Res.	Res.	Res.	BCDU [2:0]			BIN [1:0]		INIT	INITF	RSF	INITS	SHPF	WUT WF	Res.	Res.
	Reset value																0				0	0	0	0	0	0	0	0	0	0	1		
0x10	RTC_PRER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]							PREDIV_S[14:0]															
	Reset value										1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	RTC_WUTR	WUTOCLR[15:0]																WUT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	RTC_CR	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	ITSE	COE	O SEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCK SEL[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	RTC_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	INITPRIV	CALPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSPRIV	WUTPRIV	ALRBPRIV	ALRAPRIV
	Reset value																	0	0	0										0	0	0	0
0x24	RTC_WPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	RTC_ CALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	0	0	0	0				0	0	0	0	0	0	0	0	0
0x2C	RTC_SHIFTR	ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	RTC_TSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0

Table 485. RTC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x34	RTC_TSDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x38	RTC_TSSSR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	RTC_ALRMAR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	RTC_ ALRMASSR	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	RTC_ALRMBR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	RTC_ ALRMBSSR	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	RTC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUF	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
	Reset value																										0	0	0	0	0	0	0
0x54	RTC_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0	0	0	0	0	0	0
0x5C	RTC_SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
	Reset value																										0	0	0	0	0	0	0
0x64	RTC_ TAMPTSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMPTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5TS	TAMP4TS	TAMP3TS	TAMP2TS	TAMP1TS
	Reset value																0												0	0	0	0	0
0x68	RTC_TSIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSID[5:0]					
	Reset value																											0	0	0	0	0	0
0x70	RTC_ ALRABINR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	RTC_ ALRBBINR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

43 Tamper and backup registers (TAMP)

43.1 Introduction

The anti-tamper detection circuit is used to protect sensitive data from external attacks. 32 32-bit backup registers are retained in all low-power modes and also in V_{BAT} mode. The backup registers, as well as other secrets in the device, are protected by this anti-tamper detection circuit with 7 tamper pins and 14 internal tampers. The external tamper pins can be configured for edge detection, or level detection with or without filtering, or active tamper which increases the security level by auto checking that the tamper pins are not externally opened or shorted.

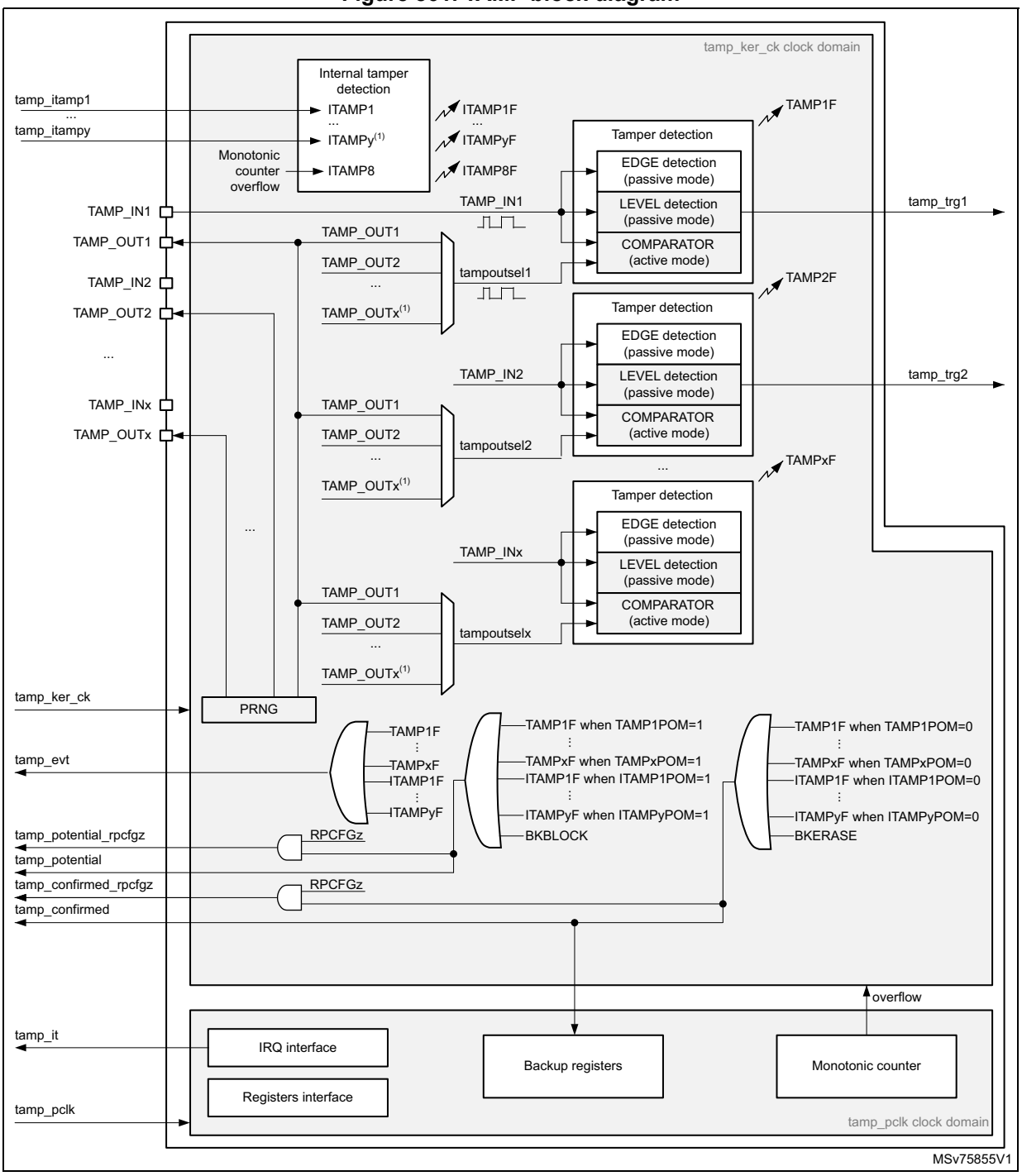
43.2 TAMP main features

- A tamper detection can optionally erase the backup registers, SRAMs, backup SRAM, HASH. The device resources protected by tamper are named “device secrets”. The list of device secrets is configurable by software.
- 32×32-bit backup registers:
  - The backup registers (TAMP_BKPxR) are implemented in the backup domain that remains powered-on by V_{BAT} when the V_{DD} power is switched off.
- Up to 7 tamper pins for 5 external tamper detection events:
  - Active tamper mode: continuous comparison between tamper output and input to protect from physical open-short attacks.
  - Flexible active tamper I/O management: from 3 meshes (each input associated to its own exclusive output) to 6 meshes (single output shared for up to 5 tamper inputs)
  - Passive tampers: ultra-low power edge or level detection with internal pull-up hardware management.
  - Configurable digital filter.
- 14 internal tamper events to protect against transient or environmental perturbation attacks
- Each tamper can be configured in two modes:
  - Confirmed mode: immediate erase of secrets on tamper detection, including backup registers erase
  - Potential mode: most of the secrets erase following a tamper detection are launched by software
- Any tamper detection can generate a RTC timestamp event.
- Tamper configuration and backup registers privilege protection
- Monotonic counter.

43.3 TAMP functional description

43.3.1 TAMP block diagram

Figure 861. TAMP block diagram



1. The number of external and internal tampers depends on products.

43.3.2 TAMP pins and internal signals

Table 486. TAMP input/output pins

Pin name	Signal type	Description
TAMP_INx (x = pin index)	Input	Tamper input pin
TAMP_OUTx (x = pin index)	Output	Tamper output pin (active mode only)

Table 487. TAMP internal input/output signals

Internal signal name	Signal type	Description
tamp_ker_ck	Input	TAMP kernel clock, connected to rtc_ker_ck and also named RTCCLK in this document
tamp_pclk	Input	TAMP APB clock, connected to rtc_pclk
tamp_itamp[y] (y = signal index)	Inputs	Internal tamper event sources
tamp_evt	Output	Tamper event detection flag (internal or external tamper), whatever confirmed or potential mode configuration.
tamp_potential	Output	Potential tamper detection signal, used for device secrets(1) protection. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in potential mode. – or a software request is done by writing BKBLOCK to 1
tamp_confirmed	Output	Confirmed tamper detection signal, used for device secrets(1) protection. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in confirmed mode. – or a software request is done by writing BKERASE to 1
tamp_potential_rpcfgz (z = signal index)	Output	Potential tamper detection signal generated only when RPCFGz = 1. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in potential mode. – or a software request is done by writing BKBLOCK to 1

Table 487. TAMP internal input/output signals (continued)

Internal signal name	Signal type	Description
tamp_confirmed_rpcfgz (z = signal index)	Output	Confirmed tamper detection signal generated only when RPCFGz = 1. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in confirmed mode. – or a software request is done by writing BKERASE to 1
tamp_it	Output	TAMP interrupt (refer to *Section 43.5: TAMP interrupts* for details)
tamp_trg[x] (x = signal index)	Output	Tamper detection trigger

1. Refer to *Table 488: TAMP interconnection*.

The TAMP kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some detections modes are not available in some low-power modes or V_{BAT} depending on the selected clock (refer to *Section 43.4: TAMP low-power modes* for more details.

Table 488. TAMP interconnection

Signal name	Source/Destination
tamp_evt	rtc_tamp_evt used to generate a timestamp event
tamp_potential	No effect.
tamp_confirmed	No effect.
tamp_potential_rpcfg0	When the bit RPCFG0 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg0 signal is used to block the read and write accesses to the device secrets listed hereafter: – Backup SRAM
tamp_confirmed_rpcfg0	When the bit RPCFG0 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg0 signal is used to erase the device secrets listed hereafter: – Backup SRAM The device secrets access is blocked when erase is on-going.
tamp_potential_rpcfg1	When the bit RPCFG1 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg1 signal is used to block the read and write access to the device secrets listed hereafter: – AXI SRAM1
tamp_confirmed_rpcfg1	When the bit RPCFG1 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg1 signal is used to erase the device secrets listed hereafter: – AXI SRAM1 The device secrets access is blocked when erase is ongoing.

Table 488. TAMP interconnection (continued)

Signal name	Source/Destination
tamp_potential_rpcfg4	When the bit RPCFG4 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg4 signal is used to erase the device secrets listed hereafter: – HASH
tamp_confirmed_rpcfg4	When the bit RPCFG4 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg4 signal is used to erase the device secrets listed hereafter – HASH
tamp_itamp1	Backup domain voltage threshold monitoring
tamp_itamp2	Temperature monitoring
tamp_itamp3	LSE monitoring (LSECSS)
tamp_itamp4	HSE monitoring (rcc_hsecss_fail)
tamp_itamp5	RTC calendar overflow (rtc_calovf)
tamp_itamp6	Unexpected debug activation
tamp_itamp7	ADC2 analog watchdog 1
tamp_itamp8(1)	Monotonic counter 1 overflow
tamp_itamp9	TRNG fault
tamp_itamp11	IWDG reset when tamper flag is set (potential tamper timeout)
tamp_itamp12	ADC2 analog watchdog 2
tamp_itamp13	ADC3 analog watchdog 1
tamp_itamp14	ADC3 analog watchdog 2
tamp_itamp15	System fault (PAR)

1. This signal is generated in the TAMP peripheral.

43.3.3 GPIOs controlled by the RTC and TAMP

The GPIOs included in the Battery Backup Domain (V_{BAT}) are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

RTC_OUT1, RTC_TS, TAMP_IN1 and TAMP_OUT2 are mapped on the same pin (PC13). The RTC and TAMP functions mapped on PC13 are available in all low-power modes and in V_{BAT} mode.

The output mechanism follows the priority order shown in *Table 489*.

Table 489. RTC pin PC13 configuration(1)

PC13 pin function		OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP2E=TAMP2AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=1	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)
TAMPALRM output Push-Pull		01 or 10 or 11	0	Don't care	Don't care	0	0	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
TAMPALRM output Open-Drain(2)	No pull	01 or 10 or 11	0	Don't care	Don't care	1	0	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
	Internal pull-up	01 or 10 or 11	0	Don't care	Don't care	1	1	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
CALIB output PP		00	0	1	0	Don't care	Don't care	Don't care	Don't care	Don't care
TAMP_OUT2 output PP		00	0	0	0	Don't care	Don't care	1	Don't care	Don't care
TAMP_IN1 input floating		00	0	0	Don't care	Don't care	Don't care	0	1	0
		00	0	1	1					
		Don't care	Don't care	0						

Table 489. RTC pin PC13 configuration(1) (continued)

PC13 pin function	OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP2E=TAMP2AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=1	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)
RTC_TS and TAMP_IN1 input floating	00	0	0	Don't care	Don't care	Don't care	0	1	1
	00	0	1	1			0		
	Don't care	Don't care	0				0		
RTC_TS input floating	00	0	0	Don't care	Don't care	Don't care	0	0	1
	00	0	1	1			0		
	Don't care	Don't care	0				0		
Wake-up pin or Standard GPIO	00	0	0	Don't care	Don't care	Don't care	0	0	0
	00	0	1	1			0		
	Don't care	Don't care	0				0		

1. OD: open drain; PP: push-pull.
2. In this configuration the GPIO must be configured in input.

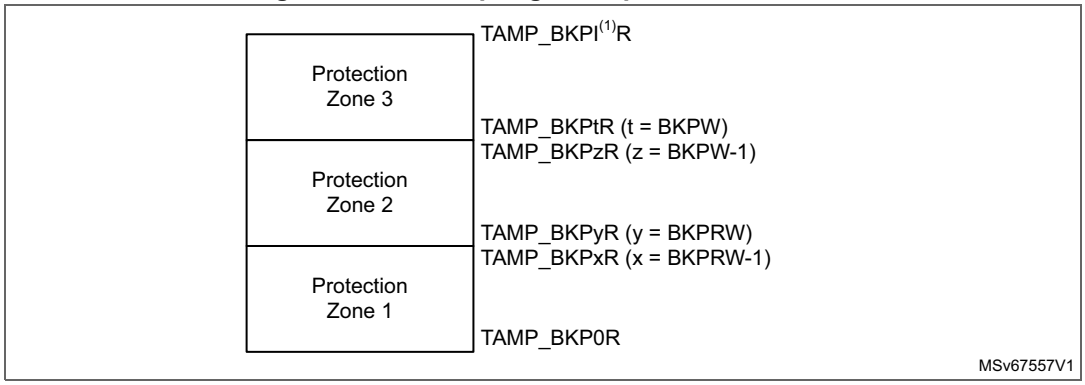
43.3.4 TAMP register write protection

After system reset, the TAMP registers (including backup registers) are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable TAMP registers write access.

43.3.5 Backup registers protection zones

The backup registers protection is configured thanks to BKPRW[7:0] and BKPW[7:0] (refer to the figure below):

Figure 862. Backup registers protection zones



1. I = last backup register index

43.3.6 TAMP privilege protection modes

By default after a backup domain power-on reset, all TAMP registers can be read or written in both privileged and non-privileged modes, except for the TAMP privilege configuration register (TAMP_PRIVCFGR) which can be written in privilege mode only. The TAMP protection configuration is not affected by a system reset.

- When the TAMPPRIV bit is set in the TAMP_PRIVCFGR register:
  - Writing the TAMP registers is possible only in privilege mode, except for the backup registers and the monotonic counter, which have their own protection settings.
  - Reading the TAMP_CFGR and TAMP_PRIVCFGR registers is always possible in both privilege and unprivileged modes. All other TAMP registers can be read only in privilege mode, except for the backup registers and the monotonic counter, which have their own protection settings.
- When the CNT1PRIV bit is set in the TAMP_PRIVCFGR register: the TAMP_COUNT1R register can be read and written only in privilege mode, and CNT1SEC can be written only in privilege mode.
- When the BKPRWPRIV bit is set in the TAMP_PRIVCFGR register: the BKPRW register can be written only in privilege mode.
- When the BKPWPRIV bit is set in the TAMP_PRIVCFGR register: the BKPRW and BKPW registers can be written only in privilege mode.

The backup registers protection is configured thanks to BKPRW[7:0] and BKPRWPRIV for the protection zone 1, and thanks to BKPRW[7:0], BKPW[7:0] and BKPWPRIV for the protection zone 2 (refer to *Figure 862*).

A non-privileged access to a privileged-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

43.3.7 Tamper detection

The tamper detection main purpose is to protect the device secrets from device external attacks. The detection is made on events on TAMP_INx (x = pin index) I/Os, or on internal monitors detecting out-of-range device conditions.

The tamper detection can be configured for the following purposes:

- erase the backup registers and other device secrets stored in SRAMs or peripherals listed in *Table 488: TAMP interconnection*. The device secrets list is configurable thanks to *TAMP resources protection configuration register (TAMP_RPCFGR)*.
- block the read/write access to the backup registers and other device secrets stored in SRAMs or peripherals listed in *Table 488: TAMP interconnection*. The device secrets list is configurable thanks to TAMP_RPCFGR.
- generate an interrupt, capable to wake-up from low-power modes
- generate a hardware trigger for the low-power timers, or a RTC timestamp event

The external I/Os tamper detection supports 2 main configurations:

- Passive mode: TAMP_INx I/Os are monitored and a tamper is detected either on edge or on level.
- Active mode: TAMP_INx (x = pin index) is continuously compared with TAMP_OUTy (y = pin index) allowing open-short detection.

A digital filter can be applied on external tamper detection to avoid false detection. In addition, it is possible to configure each tamper source in potential mode, so that the secrets erase is not launched by hardware on tamper detection. The secrets erase can then be launched by software after software checks.

43.3.8 TAMP backup registers and other device secrets erase

The backup registers (TAMP_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers and the other device secrets are not reset when the corresponding mask is set (TAMPxMSK=1 in the TAMP_CR2 register).

Note: The backup registers are also erased when the product state is changed from Closed to Open.

Tamper detection – confirmed mode

The confirmed mode is selected for TAMPx (external tamper x) when TAMPxPOM = 0 in the TAMP_CR2 register. The confirmed mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 0 in the TAMP_CR3 register. The effects of a tamper detection in confirmed mode are described with tamp_confirmed and tamp_confirmed_rpcfgx signals in the *Table 488: TAMP interconnection*.

This mode is selected to erase automatically the device secrets when the tamper is detected.

Tamper detection – potential mode

The potential mode is selected for TAMPx (external tamper x) when TAMPxPOM = 1 in the TAMP_CR2 register. The potential tamper mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 1 in the TAMP_CR3 register. The effects of a tamper detection in

potential mode are described with tamp_potential and tamp_potential_rpcfgx signals in the *Table 488: TAMP interconnection*.

This mode is selected to avoid irreversible erasure of some device secrets when the tamper is detected. In this mode, some device secrets are not erased when the corresponding tamper event is detected. In addition, the read and write accesses to these device secrets are blocked as soon as the tamper detection flag is set in potential mode, until this flag is cleared by setting the corresponding clear flag in the TAMP_SCR register. Therefore the software can perform some checks to discriminate false from true tampers, and decide to launch secrets erase only in case of the potential tamper is confirmed to be a true tamper. The device secrets are erased by software by setting the BKERASE bit in the TAMP_CR2 register.

Potential tamper to confirmed tamper timeout

Some internal tampers generate a tamper event if the independent watchdog reset occurs when another tamper flag is set (refer to *Table 488: TAMP interconnection*). The IWDG tamper must be configured with ITAMPxPOM = 0. This permits the erasure of device secrets to be forced by hardware after a timeout, in case the previous tamper event was in potential mode. This is equivalent to change the “potential tamper” into “confirmed tamper” if a watchdog reset occurs before any software decision following the potential tamper event.

Device resources protection configuration

Some device resources can be configured in order to be included to the list of the device secrets protected by tamper detection.

When RPCFGz = 0 in the TAMP_RPCFGR, the device resource associated to RPCFGz is not protected by the TAMP peripheral:

- It is not affected by tamper detection (whatever confirmed or potential mode)
- It is not affected by BKERASE software command
- It is not affected by BKBLOCK software command

When RPCFGz = 1 in the TAMP_RPCFGR, the device resource associated to RPCFGz is protected by the TAMP peripheral:

- It is affected by confirmed tamper detection and BKERASE software command, as described with tamp_confirmed_rpcfgz signal in *Table 488: TAMP interconnection*
- It is affected by potential tamper detection and BKBLOCK software command, as described with tamp_potential_rpcfgz signal in *Table 488: TAMP interconnection*

Table 490. Backup registers tamper protection

-	Potential tamper or BKBLOCK	Confirmed tamper or BKERASE
RPCFG31 = 0	No effect on backup registers	No effect on backup registers
RPCFG31 = 1	Backup registers access is blocked	Backup registers are erased

Table 491. Other device resource x tamper protection

-	Potential tamper or BKBLOCK	Confirmed tamper or BKERASE
RPCFGx = 0	No effect on device resource x	No effect on device resource x
RPCFGx = 1	Device secret x protected as described by tamp_potential_rpcfgx(1)	Device secret x protected as described by tamp_confirmed_rpcfgx(1)

1. Refer to *Table 488: TAMP interconnection*.

Device secrets access blocked by software

By default, the device secrets can be accessed by the application, except if a tamper event flag is detected: the device secrets access is not possible as long as a tamper flag is set.

It is possible to block the access to the device secrets by software, by setting the BKBLOCK bit of the TAMP_CR2 register. The device secrets access is possible only when BKBLOCK = 0 and no tamper flag is set.

43.3.9 Tamper detection configuration and initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the TAMP_CR register.

Each TAMP_INx tamper detection input is associated with a flag TAMPxF in the TAMP_SR register.

By setting the TAMPxIE bit in the TAMP_IER register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPxIE is not allowed when the corresponding TAMPxMSK is set.

Trigger output generation on tamper event

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMSK bit is cleared in TAMP_CR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMSK bit is set, the TAMPxF flag is masked, and kept cleared in TAMP_SR register. This configuration permits the low-power timers to be triggered automatically in Stop mode, without requiring the system wake-up to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

This feature is available only when the tamper is configured in the *Level detection with filtering on tamper inputs (passive mode)* mode (TAMPFLT ≠ 00 and active mode is not selected).

Timestamp on tamper event

With TAMPTS set to 1 in the RTC_CR, any internal or external tamper event causes a timestamp to occur. When TAMPTS is 0, any external tamper can be individually configured to cause or not a timestamp. A global configuration for all internal tampers is also available to enable or not timestamp on internal tamper detection (refer to *RTC timestamp on tamper control register (RTC_TAMPTSCR)*). In case a timestamp occurs due to tamper event, either the TSF bit or the TSOVF bit is set in RTC_SR, in the same manner as if a normal

timestamp event occurs. The TSID bitfield in the *RTC timestamp status register (RTC_TSIDR)* provides the identification of the cause of the timestamp event.

Caution: Timestamp on tamper event must be disabled before entering RTC initialization mode (TAMPTS, and all bits of RTC_TAMPTSCR must be cleared before setting the INIT bit in RTC_ICSR)

Note: TSF is set up to 3 ck_apre cycles after TAMPxF flags. TSF is not set if RTCCLK is stopped (it is set when RTCCLK restarts).

Note: If TAMPxF is cleared before the expected rise of TSF, TSF is not set. Consequently, in case TAMPTS = 1 or any bit in TAMP_TAMPTSCR register is set, the software should either wait for timestamp flag before clearing the tamper flag, or should read the RTC counters values in the TAMP interrupt routine.

Edge detection on tamper inputs (passive mode)

If the TAMPFLT bits are 00, the TAMP_INx pins generate tamper detection events when either a rising edge or a falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMP_INx inputs are deactivated when edge detection is selected.

Caution: When TAMPFLT = 00 and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the TAMP_INx should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (TAMP_BKPxR). This prevents the application from writing to the backup registers while the TAMP_INx input value still indicates a tamper detection. This is equivalent to a level detection on the TAMP_INx input.

Note: Tamper detection is still active when V_{DD} power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the TAMPx is mapped should be externally tied to the correct level.

Level detection with filtering on tamper inputs (passive mode)

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The TAMP_INx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the TAMP_INx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection. The TAMP_IN I/O schmitt trigger is enabled only during the precharge duration to avoid any extra consumption if the tamper switch is open (floating state).

Figure 863. Tamper sampling with precharge pulse

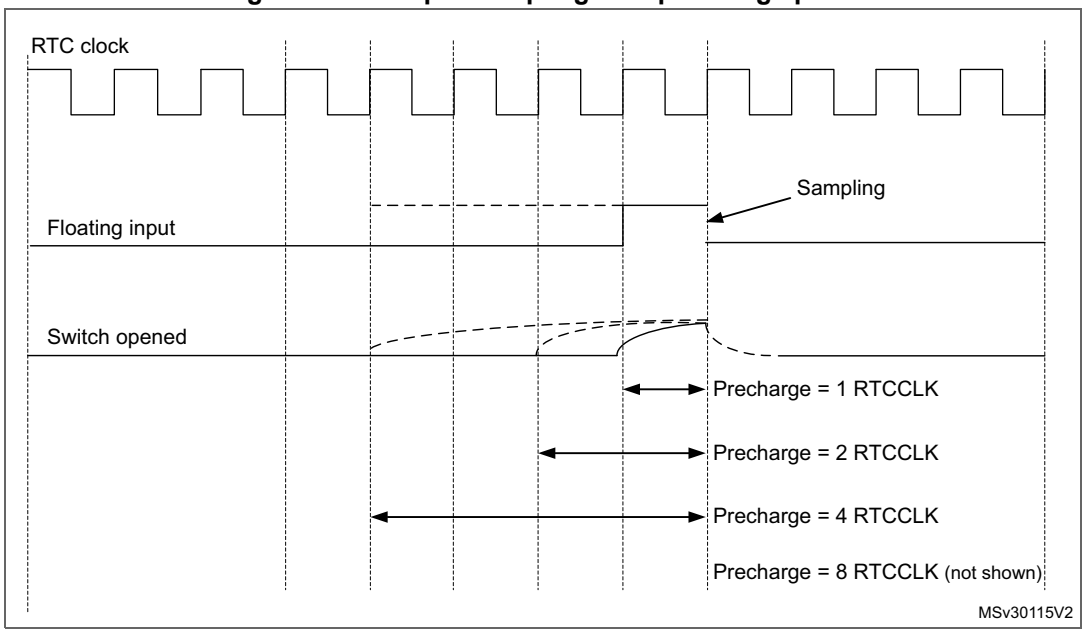
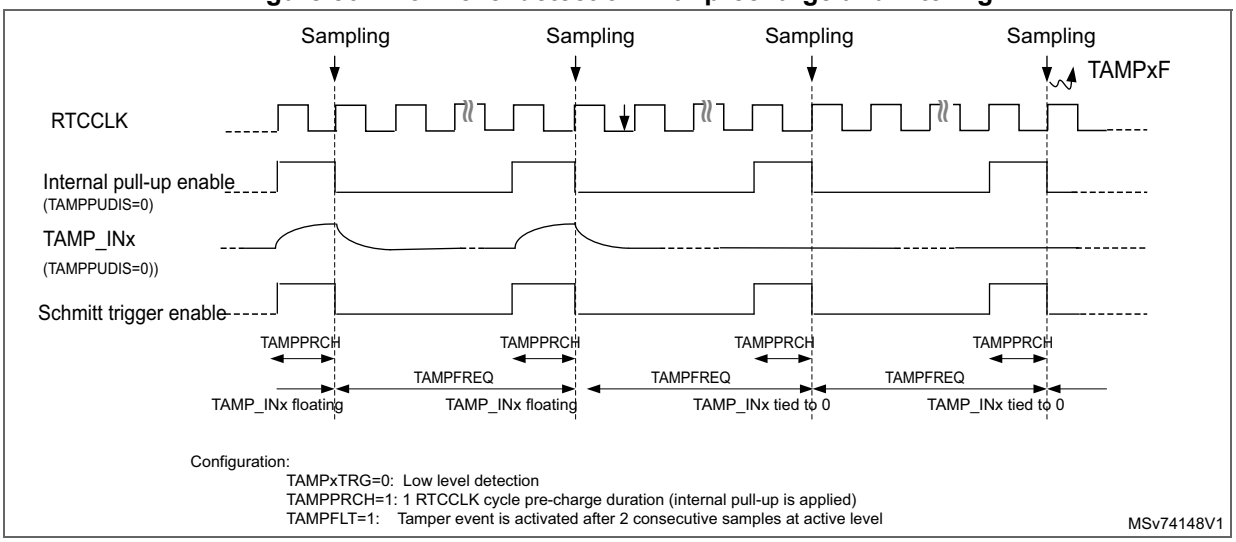


Figure 864. Low level detection with precharge and filtering



Note: Refer to the microcontroller datasheet for the electrical characteristics of the pull-up resistors.

Active tamper detection

When the TAMPxAM bit is set in the TAMP_ATCR, the tamper events are configured in active mode, which is based on a comparison between a TAMP_OUTy pin and a TAMP_INx pin. By default (ATOSHARE = 0) the comparison is made between TAMP_INx and TAMP_OUTx (y = x). When ATOSHARE bit is set, the same output can be used for several tamper inputs. The TAMP_OUTy function is enabled on the I/O as soon as it is selected for comparison with an active tamper input TAMP_INx (TAMPxEN = TAMPxAM = 1), thanks to

ATOSHARE and ATOSELx bits. Refer to ATOSHARE and ATOSEL bits descriptions in the TAMP_ATCRx (x = 1, 2) registers.

Every two CK_ATPER cycles (CK_ATPER = 2^{ATPER} × CK_ATPRE), TAMP_OUTy output pin provides a value provided by a pseudo random number generator (PRNG). After outputting this value, the TAMP_OUTy pin outputs its opposite value one CK_ATPER cycle after.

Table 492. Active tamper output change period

ATCKSEL[3:0]	CK_ATPRE frequency	ATPER[2:0]	Tamper output change (CK_ATPER) frequency	Tamper output change period(1) (ms)
0x0	f_{RTCCLK}	0x0	f_{RTCCLK}	0.030
		0x1	$f_{RTCCLK}/2$	0.061
		0x2	$f_{RTCCLK}/4$	0.122
		0x3	$f_{RTCCLK}/8$	0.244
		0x4	$f_{RTCCLK}/16$	0.488
		0x5	$f_{RTCCLK}/32$	0.977
		0x6	$f_{RTCCLK}/64$	1.953
		0x7	$f_{RTCCLK}/128$	3.906
…	…	…	…	…
0x7	$f_{RTCCLK}/128$	0x0	$f_{RTCCLK}/128$	3.906
		0x1	$f_{RTCCLK}/256$	7.8125
		0x2	$f_{RTCCLK}/512$	15.625
		0x3	$f_{RTCCLK}/1024$	31.250
		0x4	$f_{RTCCLK}/2048$	62.5
		0x5	$f_{RTCCLK}/4096$	125
		0x6	$f_{RTCCLK}/8192$	250
		0x7	$f_{RTCCLK}/16384$	500
0x8	$f_{RTCCLK}/256$	0x0	$f_{RTCCLK}/256$	7.8125
		0x1	$f_{RTCCLK}/512$	15.625
		0x2	$f_{RTCCLK}/1024$	31.250
		0x3	$f_{RTCCLK}/2048$	62.5
		0x4	$f_{RTCCLK}/4096$	125
		0x5	$f_{RTCCLK}/8192$	250
		0x6	$f_{RTCCLK}/16384$	500
		0x7	$f_{RTCCLK}/32768$	1000
…	…	…	…	…

Table 492. Active tamper output change period (continued)

ATCKSEL[3:0]	CK_ATPRE frequency	ATPER[2:0]	Tamper output change (CK_ATPER) frequency	Tamper output change period(1) (ms)
0xF	f_{RTCCLK}/32768	0x0	f_{RTCCLK}/32768	1000
		0x1	f_{RTCCLK}/65536	2000
		0x2	f_{RTCCLK}/131072	4000
		0x3	f_{RTCCLK}/262144	8000
		0x4	f_{RTCCLK}/524288	16000
		0x5	f_{RTCCLK}/1048576	32000
		0x6	f_{RTCCLK}/2097152	64000
		0x7	f_{RTCCLK}/4194304	128000

1. Assuming f_{RTCCLK} = 32768 Hz.

PRNG is consumed by the selected tamper outputs at a different frequency depending on the number of selected tamper outputs. The number of selected outputs depends on TAMPxAM, TAMPxE, ATOSEL and ATOSHARE.

- When only 1 output is selected: PRNG is consumed every 16 CK_ATPER periods.
- When 2 outputs are selected: PRNG is consumed every 8 CK_ATPER periods.
- When 3 or 4 outputs are selected: PRNG is consumed every 4 CK_ATPER periods.
- When 5 or more outputs are selected: PRNG is consumed every 2 CK_ATPER periods

The PRNG needs minimum 9 CK_ATPRE cycles to output a new value. Consequently the minimum ATPER values for correct functionality are provided in the table below:

Table 493. Minimum ATPER value

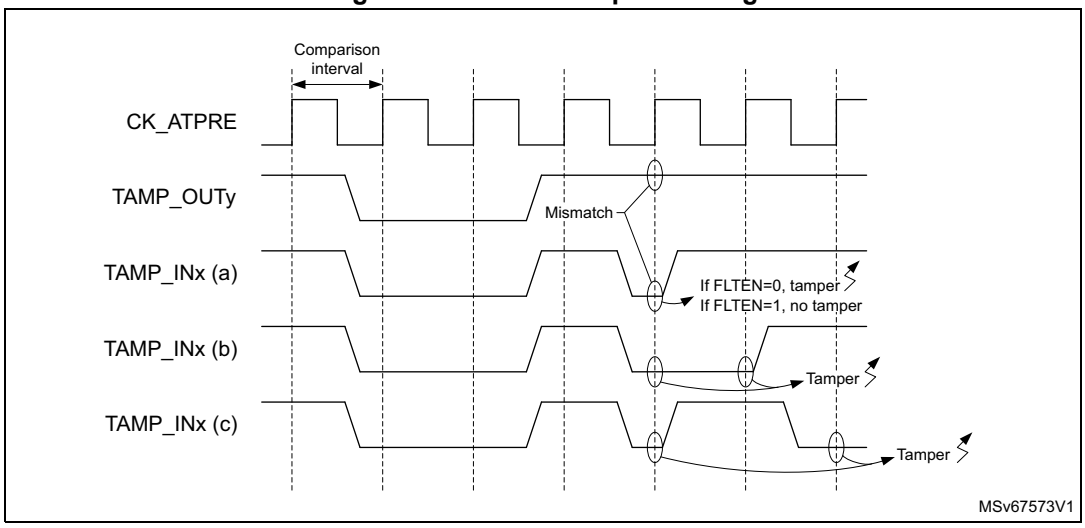
Number of selected outputs	Minimum ATPER
1	0
2	1
3 or 4	2
5 or more	3

The TAMP_INx pin is externally connected to TAMP_OUTy pin. The comparison is made between TAMP_OUTy output value and TAMP_INx received value, every CK_ATPRE cycle. In case a comparison mismatch occurs, the TAMPxF bit is set in the TAMP_SR register.

As an example, TAMP_OUT1 can be used for comparison with TAMP_IN1 and TAMP_IN2 by configuring and enabling both TAMP1 and TAMP2 in active mode, with ATOSHARE = 1, ATOSEL1 = 000 and ATOSEL2 = 000.

The active tamper can be combined with input filtering when FLTEN = 1. In this case, the tamper is detected only when 2 comparisons are false, in 4 consecutive comparison samples.

Figure 865. Active tamper filtering



As illustrated in *Figure 865*, if FLTEN = 0, any mismatch between the TAMP_OUTy output and the associated TAMP_INx input when the latter is sampled generates a tamper. This is the case in all three examples (a), (b) and (c).

If FLTEN = 1, example (a) does not generate a tamper, since only one mismatch is detected in four consecutive comparisons. In example (b), a tamper is generated since two successive mismatches are detected. Example (c) also generates a tamper, since two mismatches occur in four consecutive comparisons, even though the mismatches do not occur on successive samples.

Setting FLTEN = 1 avoids unwanted detection of tampers due to glitches, bounce or transitory states on the TAMP_INx inputs, by ignoring single pulses which are shorter than one period of CK_ATPRE, programmed in the ATCKSEL field of the TAMP_ATCR1 register. The minimum filtered pulse width is listed in *Table 494* for each possible setting of ATCKSEL, assuming f_{RTCCLK} = 32.768 kHz.

Table 494. Active tamper filtered pulse duration

ATCKSEL[3:0]	CK_ATPRE frequency	Minimum filtered pulse width (ms)
0x0	f_{RTCCLK}	0.030
0x1	$f_{RTCCLK}/2$	0.061
0x2	$f_{RTCCLK}/4$	0.122
0x3	$f_{RTCCLK}/8$	0.244
0x4	$f_{RTCCLK}/16$	0.488
0x5	$f_{RTCCLK}/32$	0.977
0x6	$f_{RTCCLK}/64$	1.953
0x7	$f_{RTCCLK}/128$	3.906
0x8	$f_{RTCCLK}/256$	7.812
0x9	$f_{RTCCLK}/512$	15.625
0xA	$f_{RTCCLK}/1024$	31.25
0xB	$f_{RTCCLK}/2048$	62.5

Table 494. Active tamper filtered pulse duration (continued)

ATCKSEL[3:0]	CK_ATPRE frequency	Minimum filtered pulse width (ms)
0xC	f_{RTCCLK}/4096	125
0xD	f_{RTCCLK}/8192	250
0xE	f_{RTCCLK}/16384	500
0xF	f_{RTCCLK}/32768	1000

Note: Multiple pulses which are shorter than one CK_ATPRE period may nevertheless cause a tamper if they result in two mismatches in four consecutive comparisons.

The pseudo-random generator must be initialized with a seed. This is done by writing consecutively four 32-bit random values in the TAMP_ATSEEDR register. Programming the seed automatically sends it to the PRNG. As long as the new seed is transferred and elaborated by the PRNG, the SEEDF bit is set in the TAMP_ATOR and it is not allowed to switch off the TAMP APB clock. The duration of the elaboration is up to 184 APB clock cycles after the forth seed is written. Consequently, after writing a new seed, the user must wait until SEEDF is cleared before entering low-power modes.

The active tamper outputs are activated only after the first seed is written and the elaboration is completed. Then new seeds can be written and elaborated during active tamper activity.

Active tamper initialization

Here is the software procedure to initialize the active tampers after system reset:

Read INITS in TAMP_ATOR register.

- If INITS = 0x0 (initialization was not done):
  a) Write TAMP_ATCR to configure Active tamper clock, filter and output sharing if any, and active mode.
  b) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  c) Write SEED by writing four times in the TAMP_ATSEEDR.
  d) Wait until SEEDF = 0 in TAMP_ATOR. Backup registers are then protected by active tamper.
- If INITS = 0x1 (initialization already done):

  No initialization. To increase randomness a new SEED should be provided regularly. When a new SEED is provided, wait until SEEDF = 0 before entering a low-power mode which switches off the TAMP APB clock.
- In case the tampers are disabled by software, and re-enabled afterwards, the SEED must be written after enabling tampers:
  a) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  b) Write SEED by writing four times in the TAMP_ATSEEDR.
  c) Wait until SEEDF = 0 in TAMP_ATOR. Backup registers are then protected by active tamper.

43.4 TAMP low-power modes

Table 495. Effect of low-power modes on TAMP

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

Table 496. TAMP pins functionality over modes

Pin name	Functional in all low-power modes	Functional in V_{BAT} mode
TAMP_IN[5:1]	Yes	No except TAMP_IN1 (PC13) and TAMP_IN2 (PI8)
TAMP_OUT[5:1]	Yes	No except TAMP_OUT2 (PC13)

43.5 TAMP interrupts

Table 497. Interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method	Exit from low-power modes
TAMP	Tamper x(2)	TAMPxF	TAMPxIE	Write 1 in CTAMPxF	Yes(3)
	Internal tamper y(2)	ITAMPyF	ITAMPyIE	Write 1 in CITAMPyF	Yes(3)

1. The event flags are in the TAMP_SR register.
2. The number of tampers and internal tampers events depend on products.
3. Refer to *Table 495: Effect of low-power modes on TAMP* for more details about available features in the low-power modes.

43.6 TAMP registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by words (32-bit).

43.6.1 TAMP control register 1 (TAMP_CR1)

This register can be protected against non-privileged access. Refer to *Section 43.3.6:*

TAMP privilege protection modes.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ITAMP15E	ITAMP14E	ITAMP13E	ITAMP12E	ITAMP11E	Res.	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	ITAMP4E	ITAMP3E	ITAMP2E	ITAMP1E
	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5E	TAMP4E	TAMP3E	TAMP2E	TAMP1E
											rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15E**: Internal tamper 15 enable
0: Internal tamper 15 disabled.
1: Internal tamper 15 enabled.

Bit 29 **ITAMP14E**: Internal tamper 14 enable
0: Internal tamper 14 disabled.
1: Internal tamper 14 enabled.

Bit 28 **ITAMP13E**: Internal tamper 13 enable
0: Internal tamper 13 disabled.
1: Internal tamper 13 enabled.

Bit 27 **ITAMP12E**: Internal tamper 12 enable
0: Internal tamper 12 disabled.
1: Internal tamper 12 enabled.

Bit 26 **ITAMP11E**: Internal tamper 11 enable
0: Internal tamper 11 disabled.
1: Internal tamper 11 enabled.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9E**: Internal tamper 9 enable
0: Internal tamper 9 disabled.
1: Internal tamper 9 enabled.

Bit 23 **ITAMP8E**: Internal tamper 8 enable
0: Internal tamper 8 disabled.
1: Internal tamper 8 enabled.

Bit 22 **ITAMP7E**: Internal tamper 7 enable
0: Internal tamper 7 disabled.
1: Internal tamper 7 enabled

Bit 21 **ITAMP6E**: Internal tamper 6 enable
0: Internal tamper 6 disabled.
1: Internal tamper 6 enabled.

Bit 20 **ITAMP5E**: Internal tamper 5 enable
0: Internal tamper 5 disabled.
1: Internal tamper 5 enabled.

Bit 19 **ITAMP4E**: Internal tamper 4 enable
0: Internal tamper 4 disabled.
1: Internal tamper 4 enabled.

Bit 18 **ITAMP3E**: Internal tamper 3 enable
0: Internal tamper 3 disabled.
1: Internal tamper 3 enabled.

Bit 17 **ITAMP2E**: Internal tamper 2 enable
0: Internal tamper 2 disabled.
1: Internal tamper 2 enabled.

Bit 16 **ITAMP1E**: Internal tamper 1 enable
0: Internal tamper 1 disabled.
1: Internal tamper 1 enabled.

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **TAMP5E**: Tamper detection on TAMP_IN5 enable(1)
0: Tamper detection on TAMP_IN5 is disabled.
1: Tamper detection on TAMP_IN5 is enabled.

Bit 3 **TAMP4E**: Tamper detection on TAMP_IN4 enable(1)
0: Tamper detection on TAMP_IN4 is disabled.
1: Tamper detection on TAMP_IN4 is enabled.

Bit 2 **TAMP3E**: Tamper detection on TAMP_IN3 enable(1)
0: Tamper detection on TAMP_IN3 is disabled.
1: Tamper detection on TAMP_IN3 is enabled.

Bit 1 **TAMP2E**: Tamper detection on TAMP_IN2 enable(1)
0: Tamper detection on TAMP_IN2 is disabled.
1: Tamper detection on TAMP_IN2 is enabled.

Bit 0 **TAMP1E**: Tamper detection on TAMP_IN1 enable(1)
0: Tamper detection on TAMP_IN1 is disabled.
1: Tamper detection on TAMP_IN1 is enabled.

1. Tamper detection mode (selected with TAMP_FLTCR, TAMP_ATCR1, TAMP_ATCR2 registers and TAMPxTRG bits in TAMP_CR2), must be configured before enabling the tamper detection.

43.6.2 TAMP control register 2 (TAMP_CR2)

This register can be protected against non-privileged access. Refer to *Section 43.3.6:*

TAMP privilege protection modes.

Address offset: 0x04

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TAMP5 TRG	TAMP4 TRG	TAMP3 TRG	TAMP2 TRG	TAMP1 TRG	BK ERASE	BK BLOCK	Res.	Res.	Res.	TAMP3 MSK	TAMP2 MSK	TAMP1 MSK
			rw	rw	rw	rw	rw	w	rw				rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5 POM	TAMP4 POM	TAMP3 POM	TAMP2 POM	TAMP1 POM
											rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **TAMP5TRG**: Active level for tamper 5 input (active mode disabled)

0: If TAMPFLT ≠ 00 tamper 5 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 5 input rising edge triggers a tamper detection event.

1: If TAMPFLT ≠ 00 tamper 5 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 5 input falling edge triggers a tamper detection event.

Bit 27 **TAMP4TRG**: Active level for tamper 4 input (active mode disabled)

0: If TAMPFLT ≠ 00 tamper 4 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 4 input rising edge triggers a tamper detection event.

1: If TAMPFLT ≠ 00 tamper 4 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 4 input falling edge triggers a tamper detection event.

Bit 26 **TAMP3TRG**: Active level for tamper 3 input

0: If TAMPFLT ≠ 00 tamper 3 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 3 input rising edge triggers a tamper detection event.

1: If TAMPFLT ≠ 00 tamper 3 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 3 input falling edge triggers a tamper detection event.

Bit 25 **TAMP2TRG**: Active level for tamper 2 input

0: If TAMPFLT ≠ 00 tamper 2 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 2 input rising edge triggers a tamper detection event.

1: If TAMPFLT ≠ 00 tamper 2 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 2 input falling edge triggers a tamper detection event.

Bit 24 **TAMP1TRG**: Active level for tamper 1 input

0: If TAMPFLT ≠ 00 tamper 1 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 1 input rising edge triggers a tamper detection event.

1: If TAMPFLT ≠ 00 tamper 1 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 1 input falling edge triggers a tamper detection event.

Bit 23 **BKERASE**: Backup registers and device secrets(1) erase

Writing ‘1’ to this bit reset the backup registers and device secrets(1). Writing 0 has no effect. This bit is always read as 0.

Bit 22 **BKBLOCK**: Backup registers and device secrets(1) access blocked

0: backup registers and device secrets(1) can be accessed if no tamper flag is set

1: backup registers and device secrets(1) cannot be accessed

Bits 21:19 Reserved, must be kept at reset value.

Bit 18 **TAMP3MSK**: Tamper 3 mask

0: Tamper 3 event generates a trigger event and TAMP3F must be cleared by software to allow next tamper event detection.

1: Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.

The tamper 3 interrupt must not be enabled when TAMP3MSK is set.

Bit 17 **TAMP2MSK**: Tamper 2 mask

0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.

1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.

The tamper 2 interrupt must not be enabled when TAMP2MSK is set.

Bit 16 **TAMP1MSK**: Tamper 1 mask

0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.

1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.

The tamper 1 interrupt must not be enabled when TAMP1MSK is set.

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **TAMP5POM**: Tamper 5 potential mode

0: Tamper 5 event detection is in confirmed mode(1).

1: Tamper 5 event detection is in potential mode(2).

Bit 3 **TAMP4POM**: Tamper 4 potential mode

0: Tamper 4 event detection is in confirmed mode(1).

1: Tamper 4 event detection is in potential mode(2).

Bit 2 **TAMP3POM**: Tamper 3 potential mode

0: Tamper 3 event detection is in confirmed mode(1).

1: Tamper 3 event detection is in potential mode(2).

Bit 1 **TAMP2POM**: Tamper 2 potential mode

0: Tamper 2 event detection is in confirmed mode(1).

1: Tamper 2 event detection is in potential mode(2).

Bit 0 **TAMP1POM**: Tamper 1 potential mode

0: Tamper 1 event detection is in confirmed mode(1).

1: Tamper 1 event detection is in potential mode(2).

1. The effects of tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in *Table 488: TAMP interconnection*.
2. The effects of tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in *Table 488: TAMP interconnection*.

43.6.3 TAMP control register 3 (TAMP_CR3)

This register can be protected against non-privileged access. Refer to *Section 43.3.6: TAMP privilege protection modes*.

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ITAMP 15POM	ITAMP 14POM	ITAMP 13POM	ITAMP 12POM	ITAMP 11POM	Res.	ITAMP9 POM	ITAMP8 POM	ITAMP7 POM	ITAMP6 POM	ITAMP5 POM	ITAMP4 POM	ITAMP3 POM	ITAMP2 POM	ITAMP1 POM
	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **ITAMP15POM**: Internal tamper 15 potential mode
0: Internal tamper 15 event detection is in confirmed mode(1).
1: Internal tamper 15 event detection is in potential mode(2).

Bit 13 **ITAMP14POM**: Internal Tamper 14 potential mode
0: Internal tamper 14 event detection is in confirmed mode(1).
1: Internal tamper 14 event detection is in potential mode(2).

Bit 12 **ITAMP13POM**: Internal tamper 13 potential mode
0: Internal tamper 13 event detection is in confirmed mode(1).
1: Internal tamper 13 event detection is in potential mode(2).

Bit 11 **ITAMP12POM**: Internal tamper 12 potential mode
0: Internal tamper 12 event detection is in confirmed mode(1).
1: Internal tamper 12 event detection is in potential mode(2).

Bit 10 **ITAMP11POM**: Internal tamper 11 potential mode
0: Internal tamper 11 event detection is in confirmed mode(1).
1: Internal tamper 11 event detection is in potential mode(2).

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ITAMP9POM**: Internal tamper 9 potential mode
0: Internal tamper 9 event detection is in confirmed mode(1).
1: Internal tamper 9 event detection is in potential mode(2).

Bit 7 **ITAMP8POM**: Internal tamper 8 potential mode
0: Internal tamper 8 event detection is in confirmed mode(1).
1: Internal tamper 8 event detection is in potential mode(2).

Bit 6 **ITAMP7POM**: Internal tamper 7 potential mode
0: Internal tamper 7 event detection is in confirmed mode(1).
1: Internal tamper 7 event detection is in potential mode(2).

Bit 5 **ITAMP6POM**: Internal tamper 6 potential mode
0: Internal tamper 6 event detection is in confirmed mode(1).
1: Internal tamper 6 event detection is in potential mode(2).

Bit 4 **ITAMP5POM**: Internal tamper 5 potential mode
0: Internal tamper 5 event detection is in confirmed mode(1).
1: Internal tamper 5 event detection is in potential mode(2).

Bit 3 **ITAMP4POM**: Internal tamper 4 potential mode
0: Internal tamper 4 event detection is in confirmed mode(1).
1: Internal tamper 4 event detection is in potential mode(2).

Bit 2 **ITAMP3POM**: Internal tamper 3 potential mode
0: Internal tamper 3 event detection is in confirmed mode(1).
1: Internal tamper 3 event detection is in potential mode(2).

Bit 1 **ITAMP2POM**: Internal tamper 2 potential mode
0: Internal tamper 2 event detection is in confirmed mode(1).
1: Internal tamper 2 event detection is in potential mode(2).

Bit 0 **ITAMP1POM**: Internal tamper 1 potential mode
0: Internal tamper 1 event detection is in confirmed mode(1).
1: Internal tamper 1 event detection is in potential mode(2).

1. The effects of internal tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in *Table 488: TAMP interconnection*
2. The effects of internal tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in *Table 488: TAMP interconnection*.

43.6.4 TAMP filter control register (TAMP_FLTCR)

This register can be protected against non-privileged access. Refer to *Section 43.3.6: TAMP privilege protection modes*.

Address offset: 0x0C

Backup domain reset value: 0x0000 0000

System reset: not affected

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

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

Bit 7 **TAMPPUDIS**: TAMP_INx pull-up disable

This bit determines if each of the TAMPx pins are precharged before each sample.

0: Precharge TAMP_INx pins before sampling (enable internal pull-up)

1: Disable precharge of TAMP_INx pins.

Bits 6:5 **TAMPPRCH[1:0]**: TAMP_INx precharge duration

These bit determines the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the TAMP_INx inputs.

0x0: 1 RTCCLK cycle

0x1: 2 RTCCLK cycles

0x2: 4 RTCCLK cycles

0x3: 8 RTCCLK cycles

Bits 4:3 **TAMPFLT[1:0]**: TAMP_INx filter count

These bits determines the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a tamper event. TAMPFLT is valid for each of the TAMP_INx inputs.

0x0: Tamper event is activated on edge of TAMP_INx input transitions to the active level (no internal pull-up on TAMP_INx input).

0x1: Tamper event is activated after 2 consecutive samples at the active level.

0x2: Tamper event is activated after 4 consecutive samples at the active level.

0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 2:0 **TAMPFREQ[2:0]**: Tamper sampling frequency

Determines the frequency at which each of the TAMP_INx inputs are sampled.

0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)

0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)

0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)

0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)

0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)

0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)

0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)

0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

Note: This register concerns only the tamper inputs in passive mode.

43.6.5 TAMP active tamper control register 1 (TAMP_ATCR1)

This register can be protected against non-privileged access. Refer to *Section 43.3.6: TAMP privilege protection modes*.

Address offset: 0x10

Backup domain reset value: 0x0007 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTEN	ATO SHARE	Res.	Res.	Res.	ATPER[2:0]			Res.	Res.	Res.	Res.	ATCKSEL[3:0]			
rw	rw				rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5 AM	TAMP4 AM	TAMP3 AM	TAMP2 AM	TAMP1 AM
											rw	rw	rw	rw	rw

Bit 31 **FLTEN**: Active tamper filter enable
0: Active tamper filtering disable
1: Active tamper filtering enable

Bit 30 **ATOSHARE**: Active tamper output sharing
0: Each active tamper input TAMP_INi is compared with its dedicated output TAMP_OUTi
1: Each active tamper input TAMP_INi is compared with TAMPOUTSELi defined by ATOSELi bits.

Bits 29:27 Reserved, must be kept at reset value.

Bits 26:24 **ATPER[2:0]**: Active tamper output change period
The tamper output is changed every CK_ATPER = (2^{ATPER} x CK_ATPRE) cycles. Refer to *Table 493: Minimum ATPER value*.

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

Bits 19:16 **ATCKSEL[3:0]**: Active tamper RTC asynchronous prescaler clock selection
These bits selects the RTC asynchronous prescaler stage output. The selected clock is CK_ATPRE = RTCCLK/$2^{ATCKSEL}$.

Note: These bits can be written only when all active tampers are disabled. The write protection remains for up to 1.5 CK_ATPRE cycles after all the active tampers are disable.

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **TAMP5AM**: Tamper 5 active mode
0: Tamper 5 detection mode is passive.
1: Tamper 5 detection mode is active.

Bit 3 **TAMP4AM**: Tamper 4 active mode
0: Tamper 4 detection mode is passive.
1: Tamper 4 detection mode is active.

Bit 2 **TAMP3AM**: Tamper 3 active mode
0: Tamper 3 detection mode is passive.
1: Tamper 3 detection mode is active.

Bit 1 **TAMP2AM**: Tamper 2 active mode
0: Tamper 2 detection mode is passive.
1: Tamper 2 detection mode is active.

Bit 0 **TAMP1AM**: Tamper 1 active mode
0: Tamper 1 detection mode is passive.
1: Tamper 1 detection mode is active.

Note: Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

43.6.6 TAMP active tamper seed register (TAMP_ATSEEDR)

This register can be protected against non-privileged access. Refer to *Section 43.3.6: TAMP privilege protection modes*.

Address offset: 0x14

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **SEED[31:0]**: Pseudo-random generator seed value
This register must be written four times with 32-bit values to provide the 128-bit seed to the PRNG. Writing to this register automatically sends the seed value to the PRNG.

43.6.7 TAMP active tamper output register (TAMP_ATOR)

This register can be protected against non-privileged access. Refer to *Section 43.3.6: TAMP privilege protection modes*.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected, except for SEEDF which is reset to 0.

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

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **INITS**: Active tamper initialization status

This flag is set by hardware when the PRNG has absorbed the first 128-bit seed, meaning that the enabled active tampers are functional. This flag is cleared when the active tampers are disabled.

Bit 14 **SEEDF**: Seed running flag

This flag is set by hardware when a new seed is written in the TAMP_ATSEEDR. It is cleared by hardware when the PRNG has absorbed this new seed, and by system reset. The TAMP APB clock must not be switched off as long as SEEDF is set.

Bits 13:8 Reserved, must be kept at reset value.

Bits 7:0 **PRNG[7:0]**: Pseudo-random generator value

This field provides the values of the PRNG output. Because of potential inconsistencies due to synchronization delays, PRNG must be read at least twice. The read value is correct if it is equal to previous read value.

43.6.8 TAMP active tamper control register 2 (TAMP_ATCR2)

This register can be protected against non-privileged access. Refer to *Section 43.3.6: TAMP privilege protection modes*.

Address offset: 0x1C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATOSEL5[2:0]			ATOSEL4[2:0]			ATOSEL3[2]
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATOSEL3[1:0]		ATOSEL2[2:0]			ATOSEL1[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

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

Bits 22:20 **ATOSEL5[2:0]**: Active tamper shared output 5 selection
000: TAMPOUTSEL5 = TAMP_OUT1
001: TAMPOUTSEL5 = TAMP_OUT2
010: TAMPOUTSEL5 = TAMP_OUT3
011: TAMPOUTSEL5 = TAMP_OUT4
100: TAMPOUTSEL5 = TAMP_OUT5
101: Reserved
110: Reserved
111: Reserved
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 19:17 **ATOSEL4[2:0]**: Active tamper shared output 4 selection
000: TAMPOUTSEL4 = TAMP_OUT1
001: TAMPOUTSEL4 = TAMP_OUT2
010: TAMPOUTSEL4 = TAMP_OUT3
011: TAMPOUTSEL4 = TAMP_OUT4
100: TAMPOUTSEL4 = TAMP_OUT5
101: Reserved
110: Reserved
111: Reserved
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 16:14 **ATOSEL3[2:0]**: Active tamper shared output 3 selection
000: TAMPOUTSEL3 = TAMP_OUT1
001: TAMPOUTSEL3 = TAMP_OUT2
010: TAMPOUTSEL3 = TAMP_OUT3
011: TAMPOUTSEL3 = TAMP_OUT4
100: TAMPOUTSEL3 = TAMP_OUT5
101: Reserved
110: Reserved
111: Reserved
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 13:11 **ATOSEL2[2:0]**: Active tamper shared output 2 selection
000: TAMPOUTSEL2 = TAMP_OUT1
001: TAMPOUTSEL2 = TAMP_OUT2
010: TAMPOUTSEL2 = TAMP_OUT3
011: TAMPOUTSEL2 = TAMP_OUT4
100: TAMPOUTSEL2 = TAMP_OUT5
101: Reserved
110: Reserved
111: Reserved
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 10:8 **ATOSEL1[2:0]**: Active tamper shared output 1 selection
000: TAMPOUTSEL1 = TAMP_OUT1
001: TAMPOUTSEL1 = TAMP_OUT2
010: TAMPOUTSEL1 = TAMP_OUT3
011: TAMPOUTSEL1 = TAMP_OUT4
100: TAMPOUTSEL1 = TAMP_OUT5
101: Reserved
110: Reserved
111: Reserved
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

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

Note: Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

43.6.9 TAMP configuration register (TAMP_CFGR)

This register can be globally write-protected, or each bit of this register can be individually write-protected against non-privileged access depending on the TAMP_PRIVCFGR configuration (refer to *Section 43.3.6: TAMP privilege protection modes*).

Address offset: 0x20

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **BKPW[7:0]**: Backup registers write protection offset
BKPW value must be from 0 to 32.

Protection zone 2 is defined for backup registers from TAMP_BKPyR (y = BKPRW) to TAMP_BKPzR (z = BKPW-1, with BKPW > BKPRW):
If BKPW = 0 or if BKPW ≤ BKPRW: there is no protection zone 2.

Protection zone 3 is defined for backup registers from TAMP_BKPtR (t = BKPW if BKPW ≥ BKPRW, else t = BKPRW).
If BKPW = 32: there is no protection zone 3.

Refer to *Figure 862: Backup registers protection zones*.

Note: If BKPWPRIV is set, BKPW[7:0] can be written only in privileged mode.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **BKPRW[7:0]**: Backup registers read/write protection offset
BKPRW value must be from 0 to 32.

Protection zone 1 is defined for backup registers from TAMP_BKP0R to TAMP_BKPxR (x = BKPRW-1, with BKPRW ≥ 1).
If BKPRW = 0: there is no protection zone 1.

Refer to *Figure 862: Backup registers protection zones*.

Note: If BKPRWPRIV or BKPWPRIV is set, BKPRW[7:0] can only be written in privileged mode.

43.6.10 TAMP privilege configuration register (TAMP_PRIVCFGR)

This register can be written only when the APB access is privileged.

Address offset: 0x24

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMP PRIV	BKP WPRIV	BKPR WPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1 PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

Bit 31 **TAMPPRIV**: Tamper privilege protection (excluding monotonic counter and backup registers)
0: Tamper configuration and interrupt can be written with privileged or unprivileged access.
1: Tamper configuration and interrupt can be written only with privileged access.
Note: Refer to Section 43.3.6: TAMP privilege protection modes for details on the read protection.

Bit 30 **BKPWPRIV**: Backup registers zone 2 privilege protection
0: Backup registers zone 2 can be written with privileged or unprivileged access.
1: Backup registers zone 2 can be written only with privileged access.

Bit 29 **BKPRWPRIV**: Backup registers zone 1 privilege protection
0: Backup registers zone 1 can be read and written with privileged or unprivileged access.
1: Backup registers zone 1 can be read and written only with privileged access

Bits 28:16 Reserved, must be kept at reset value.

Bit 15 **CNT1PRIV**: Monotonic counter 1 privilege protection
0: Monotonic counter 1 (TAMP_COUNT1R) can be read and written when the APB access is privileged or non-privileged.
1: Monotonic counter 1 (TAMP_COUNT1R) can be read and written only when the APB access is privileged.

Bits 14:0 Reserved, must be kept at reset value.

43.6.11 TAMP interrupt enable register (TAMP_IER)

This register can be protected against non-privileged access. Refer to *Section 43.3.6: TAMP privilege protection modes*.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ITAMP15IE	ITAMP14IE	ITAMP13IE	ITAMP12IE	ITAMP11IE	Res.	ITAMP9IE	ITAMP8IE	ITAMP7IE	ITAMP6IE	ITAMP5IE	ITAMP4IE	ITAMP3IE	ITAMP2IE	ITAMP1IE
	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5IE	TAMP4IE	TAMP3IE	TAMP2IE	TAMP1IE
											rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15IE**: Internal tamper 15 interrupt enable
0: Internal tamper 15 interrupt disabled.
1: Internal tamper 15 interrupt enabled.

Bit 29 **ITAMP14IE**: Internal tamper 14 interrupt enable
0: Internal tamper 14 interrupt disabled.
1: Internal tamper 14 interrupt enabled.

Bit 28 **ITAMP13IE**: Internal tamper 13 interrupt enable
0: Internal tamper 13 interrupt disabled.
1: Internal tamper 13 interrupt enabled.

Bit 27 **ITAMP12IE**: Internal tamper 12 interrupt enable
0: Internal tamper 12 interrupt disabled.
1: Internal tamper 12 interrupt enabled.

Bit 26 **ITAMP11IE**: Internal tamper 11 interrupt enable
0: Internal tamper 11 interrupt disabled.
1: Internal tamper 11 interrupt enabled.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9IE**: Internal tamper 9 interrupt enable
0: Internal tamper 9 interrupt disabled.
1: Internal tamper 9 interrupt enabled.

Bit 23 **ITAMP8IE**: Internal tamper 8 interrupt enable
0: Internal tamper 8 interrupt disabled.
1: Internal tamper 8 interrupt enabled.

Bit 22 **ITAMP7IE**: Internal tamper 7 interrupt enable
0: Internal tamper 7 interrupt disabled.
1: Internal tamper 7 interrupt enabled.

Bit 21 **ITAMP6IE**: Internal tamper 6 interrupt enable
0: Internal tamper 6 interrupt disabled.
1: Internal tamper 6 interrupt enabled.

Bit 20 **ITAMP5IE**: Internal tamper 5 interrupt enable
0: Internal tamper 5 interrupt disabled.
1: Internal tamper 5 interrupt enabled.

Bit 19 **ITAMP4IE**: Internal tamper 4 interrupt enable
0: Internal tamper 4 interrupt disabled.
1: Internal tamper 4 interrupt enabled.

Bit 18 **ITAMP3IE**: Internal tamper 3 interrupt enable
0: Internal tamper 3 interrupt disabled.
1: Internal tamper 3 interrupt enabled.

Bit 17 **ITAMP2IE**: Internal tamper 2 interrupt enable
0: Internal tamper 2 interrupt disabled.
1: Internal tamper 2 interrupt enabled.

Bit 16 **ITAMP1IE**: Internal tamper 1 interrupt enable
0: Internal tamper 1 interrupt disabled.
1: Internal tamper 1 interrupt enabled

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **TAMP5IE**: Tamper 5 interrupt enable
0: Tamper 5 interrupt disabled.
1: Tamper 5 interrupt enabled.

Bit 3 **TAMP4IE**: Tamper 4 interrupt enable
0: Tamper 4 interrupt disabled.
1: Tamper 4 interrupt enabled.

Bit 2 **TAMP3IE**: Tamper 3 interrupt enable
0: Tamper 3 interrupt disabled.
1: Tamper 3 interrupt enabled..

Bit 1 **TAMP2IE**: Tamper 2 interrupt enable
0: Tamper 2 interrupt disabled.
1: Tamper 2 interrupt enabled.

Bit 0 **TAMP1IE**: Tamper 1 interrupt enable
0: Tamper 1 interrupt disabled.
1: Tamper 1 interrupt enabled.

43.6.12 TAMP status register (TAMP_SR)

This register can be protected against non-privileged access. Refer to *Section 43.3.6: TAMP privilege protection modes*.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ITAMP15F	ITAMP14F	ITAMP13F	ITAMP12F	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	ITAMP5F	ITAMP4F	ITAMP3F	ITAMP2F	ITAMP1F
	r	r	r	r	r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5F	TAMP4F	TAMP3F	TAMP2F	TAMP1F
											r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15F**: Internal tamper 15 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 15.

Bit 29 **ITAMP14F**: Internal tamper 14 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 14.

Bit 28 **ITAMP13F**: Internal tamper 13 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 13.

Bit 27 **ITAMP12F**: Internal tamper 12 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 12.

Bit 26 **ITAMP11F**: Internal tamper 11 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 11.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9F**: Internal tamper 9 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 9.

Bit 23 **ITAMP8F**: Internal tamper 8 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 8.

Bit 22 **ITAMP7F**: Internal tamper 7 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 7.

Bit 21 **ITAMP6F**: Internal tamper 6 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 6.

Bit 20 **ITAMP5F**: Internal tamper 5 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.

Bit 19 **ITAMP4F**: Internal tamper 4 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 4.

Bit 18 **ITAMP3F**: Internal tamper 3 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.

Bit 17 **ITAMP2F**: Internal tamper 2 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 2.

Bit 16 **ITAMP1F**: Internal tamper 1 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 1.

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **TAMP5F**: TAMP5 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP5 input.

Bit 3 **TAMP4F**: TAMP4 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP4 input.

Bit 2 **TAMP3F**: TAMP3 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP3 input.

Bit 1 **TAMP2F**: TAMP2 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP2 input.

Bit 0 **TAMP1F**: TAMP1 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP1 input.

43.6.13 TAMP masked interrupt status register (TAMP_MISR)

This register can be protected against non-privileged access. Refer to *Section 43.3.6: TAMP privilege protection modes*.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ITAMP15MF	ITAMP14MF	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF
	r	r	r	r	r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
											r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15MF**: internal tamper 15 interrupt masked flag
This flag is set by hardware when the internal tamper 15 interrupt is raised.

Bit 29 **ITAMP14MF**: internal tamper 14 interrupt masked flag
This flag is set by hardware when the internal tamper 14 interrupt is raised.

Bit 28 **ITAMP13MF**: internal tamper 13 interrupt masked flag
This flag is set by hardware when the internal tamper 13 interrupt is raised.

Bit 27 **ITAMP12MF**: internal tamper 12 interrupt masked flag
This flag is set by hardware when the internal tamper 12 interrupt is raised.

Bit 26 **ITAMP11MF**: internal tamper 11 interrupt masked flag
This flag is set by hardware when the internal tamper 11 interrupt is raised.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9MF**: internal tamper 9 interrupt masked flag
This flag is set by hardware when the internal tamper 9 interrupt is raised.

Bit 23 **ITAMP8MF**: Internal tamper 8 interrupt masked flag
This flag is set by hardware when the internal tamper 8 interrupt is raised.

Bit 22 **ITAMP7MF**: Internal tamper 7 tamper interrupt masked flag
This flag is set by hardware when the internal tamper 7 interrupt is raised.

Bit 21 **ITAMP6MF**: Internal tamper 6 interrupt masked flag
This flag is set by hardware when the internal tamper 6 interrupt is raised.

Bit 20 **ITAMP5MF**: Internal tamper 5 interrupt masked flag
This flag is set by hardware when the internal tamper 5 interrupt is raised.

Bit 19 **ITAMP4MF**: Internal tamper 4 interrupt masked flag
This flag is set by hardware when the internal tamper 4 interrupt is raised.

Bit 18 **ITAMP3MF**: Internal tamper 3 interrupt masked flag
This flag is set by hardware when the internal tamper 3 interrupt is raised.

Bit 17 **ITAMP2MF**: Internal tamper 2 interrupt masked flag
This flag is set by hardware when the internal tamper 2 interrupt is raised.

Bit 16 **ITAMP1MF**:Internal tamper 1 interrupt masked flag
This flag is set by hardware when the internal tamper 1 interrupt is raised.

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **TAMP5MF**: TAMP5 interrupt masked flag
This flag is set by hardware when the tamper 5 interrupt is raised.

Bit 3 **TAMP4MF**: TAMP4 interrupt masked flag
This flag is set by hardware when the tamper 4 interrupt is raised.

Bit 2 **TAMP3MF**: TAMP3 interrupt masked flag
This flag is set by hardware when the tamper 3 interrupt is raised.

Bit 1 **TAMP2MF**: TAMP2 interrupt masked flag
This flag is set by hardware when the tamper 2 interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 interrupt masked flag
This flag is set by hardware when the tamper 1 interrupt is raised.

43.6.14 TAMP status clear register (TAMP_SCR)

This register can be protected against non-privileged access. Refer to *Section 43.3.6: TAMP privilege protection modes*.

Address offset: 0x3C

System reset value: 0x0000 0000

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CITAMP15F**: Clear ITAMP15 detection flag
Writing 1 in this bit clears the ITAMP15F bit in the TAMP_SR register.

Bit 29 **CITAMP14F**: Clear ITAMP14 detection flag
Writing 1 in this bit clears the ITAMP14F bit in the TAMP_SR register.

Bit 28 **CITAMP13F**: Clear ITAMP13 detection flag
Writing 1 in this bit clears the ITAMP13F bit in the TAMP_SR register.

Bit 27 **CITAMP12F**: Clear ITAMP12 detection flag
Writing 1 in this bit clears the ITAMP12F bit in the TAMP_SR register.

Bit 26 **CITAMP11F**: Clear ITAMP11 detection flag
Writing 1 in this bit clears the ITAMP11F bit in the TAMP_SR register.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **CITAMP9F**: Clear ITAMP9 detection flag
Writing 1 in this bit clears the ITAMP9F bit in the TAMP_SR register.

Bit 23 **CITAMP8F**: Clear ITAMP8 detection flag
Writing 1 in this bit clears the ITAMP8F bit in the TAMP_SR register.

Bit 22 **CITAMP7F**: Clear ITAMP7 detection flag
Writing 1 in this bit clears the ITAMP7F bit in the TAMP_SR register.

Bit 21 **CITAMP6F**: Clear ITAMP6 detection flag
Writing 1 in this bit clears the ITAMP6F bit in the TAMP_SR register.

Bit 20 **CITAMP5F**: Clear ITAMP5 detection flag
Writing 1 in this bit clears the ITAMP5F bit in the TAMP_SR register.

Bit 19 **CITAMP4F**: Clear ITAMP4 detection flag
Writing 1 in this bit clears the ITAMP4F bit in the TAMP_SR register.

Bit 18 **CITAMP3F**: Clear ITAMP3 detection flag
Writing 1 in this bit clears the ITAMP3F bit in the TAMP_SR register.

Bit 17 **CITAMP2F**: Clear ITAMP2 detection flag
Writing 1 in this bit clears the ITAMP2F bit in the TAMP_SR register.

Bit 16 **CITAMP1F**: Clear ITAMP1 detection flag
Writing 1 in this bit clears the ITAMP1F bit in the TAMP_SR register.

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **CTAMP5F**: Clear TAMP5 detection flag
Writing 1 in this bit clears the TAMP5F bit in the TAMP_SR register.

Bit 3 **CTAMP4F**: Clear TAMP4 detection flag
Writing 1 in this bit clears the TAMP4F bit in the TAMP_SR register.

Bit 2 **CTAMP3F**: Clear TAMP3 detection flag
Writing 1 in this bit clears the TAMP3F bit in the TAMP_SR register.

Bit 1 **CTAMP2F**: Clear TAMP2 detection flag
Writing 1 in this bit clears the TAMP2F bit in the TAMP_SR register.

Bit 0 **CTAMP1F**: Clear TAMP1 detection flag
Writing 1 in this bit clears the TAMP1F bit in the TAMP_SR register.

43.6.15 TAMP monotonic counter 1 register (TAMP_COUNT1R)

This register can be protected against non-privileged access. Refer to *Section 43.3.6: TAMP privilege protection modes*.

Address offset: 0x040

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **COUNT[31:0]:**

The monotonic counter increments by one when a write access occurs to this register, regardless of the written value. Reading this register returns the monotonic counter value. This register does not roll over and freezes upon reaching its maximum value.

43.6.16 TAMP resources protection configuration register (TAMP_RPCFGR)

This register can be protected against non-privileged access. Refer to *Section 43.3.6: TAMP privilege protection modes*.

Address offset: 0x54

Backup domain reset value: 0x8000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RP CFG31	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RP CFG4	Res.	Res.	RP CFG1	RP CFG0
											rw			rw	rw

Bit 31 **RPCFG31**: Backup registers protection

0: Backup registers and BHKLOCK are not included in the device secrets protected by TAMP peripheral

1: Backup registers and BHKLOCK are included in the device secrets protected by TAMP peripheral

Bits 30:5 Reserved, must be kept at reset value.

Bit 4 **RPCFG4**: Configurable resource 4 protection (1)

0: Resource y is not included in the device secrets protected by TAMP peripheral

1: Resource y is included in the device secrets protected by TAMP peripheral

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **RPCFG1**: Configurable resource 1 protection (1)

0: Resource y is not included in the device secrets protected by TAMP peripheral
1: Resource y is included in the device secrets protected by TAMP peripheral

Bit 0 **RPCFG0**: Configurable resource 0 protection (1)

0: Resource y is not included in the device secrets protected by TAMP peripheral
1: Resource y is included in the device secrets protected by TAMP peripheral

1. Refer to tamp_confirmed_rpcfg*y* and tamp_potential_rpcfg*y* (y= 0, 1, 4) signals in *Table 486: TAMP input/output pins* and *Table 488: TAMP interconnection*.

43.6.17 TAMP backup x register (TAMP_BKPxR)

This register can be protected against non-privileged access. Refer to *Section 43.3.6: TAMP privilege protection modes*.

Address offset: 0x100 + 0x04 * x, (x = 0 to 31)

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **BKP[31:0]:**

The application can write or read data to and from these registers.
In the default (ERASE) configuration this register is reset on a tamper detection event. It is forced to reset value as long as there is at least one internal or external tamper flag being set. This register is also reset when the product state is opened.

43.6.18 TAMP register map

Table 498. TAMP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TAMP_CR1	Res.	ITAMP15E	ITAMP14E	ITAMP13E	ITAMP12E	ITAMP11E	Res.	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	ITAMP4E	ITAMP3E	ITAMP2E	ITAMP1E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5E	TAMP4E	TAMP3E	TAMP2E	TAMP1E
	Reset value		0	0	0	0	0		0	0	0	0	0	0	0	0	0												0	0	0	0	0
0x04	TAMP_CR2	Res.	Res.	Res.	TAMP5TRG	TAMP4TRG	TAMP3TRG	TAMP2TRG	TAMP1TRG	BKERASE	BKBLOCK	Res.	Res.	Res.	TAMP3MSK	TAMP2MSK	TAMP1MSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5POM	TAMP4POM	TAMP3POM	TAMP2POM	TAMP1POM
	Reset value				0	0	0	0	0	0	0				0	0	0												0	0	0	0	0
0x08	TAMP_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP15POM	ITAMP14POM	ITAMP13POM	ITAMP12POM	ITAMP11POM	Res.	ITAMP9POM	ITAMP8POM	ITAMP7POM	ITAMP6POM	ITAMP5POM	ITAMP4POM	ITAMP3POM	ITAMP2POM	ITAMP1POM
	Reset value																		0	0	0	0	0		0	0	0	0	0	0	0	0	0
0x0C	TAMP_FLTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]		
	Reset value																									0	0	0	0	0	0	0	0
0x10	TAMP_ATCR1	FLTEN	ATOSHARE	Res.	Res.	Res.	AT PER[2:0]			Res.	Res.	Res.	Res.	ATCK SEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5AM	TAMP4AM	TAMP3AM	TAMP2AM	TAMP1AM
	Reset value	0	0				0	0	0					0	1	1	1												0	0	0	0	0
0x14	TAMP_ ATSEEDR	SEED[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	TAMP_ATOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INITS	SEEDF	Res.	Res.	Res.	Res.	Res.	Res.	PRNG[7:0]							
	Reset value																	0	0							0	0	0	0	0	0	0	0
0x1C	TAMP_ATCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATO SEL5 [2:0]			ATO SEL4 [2:0]			ATO SEL3 [2:0]			ATO SEL2 [2:0]			ATO SEL1 [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x20	TAMP_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRW[7:0]							
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x24	TAMP_ PRIVCFGR	TAMPPRIV	BKPWPRIV	BKPRWPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT1PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0														0															
0x2C	TAMP_IER	Res.	ITAMP15IE	ITAMP14IE	ITAMP13IE	ITAMP12IE	ITAMP11IE	Res.	ITAMP9IE	ITAMP8IE	ITAMP7IF	ITAMP6IE	ITAMP5IE	ITAMP4IE	ITAMP3IE	ITAMP2IE	ITAMP1IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5IE	TAMP4IE	TAMP3IE	TAMP2IE	TAMP1IE
	Reset value		0	0	0	0	0		0	0	0	0	0	0	0	0	0												0	0	0	0	0

Table 498. TAMP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	TAMP_SR	Res.	ITAMP15F	ITAMP14F	ITAMP13F	ITAMP12F	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	ITAMP5F	ITAMP4F	ITAMP3F	ITAMP2F	ITAMP1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5F	TAMP4F	TAMP3F	TAMP2F	TAMP1F
	Reset value		0	0	0	0	0		0	0	0	0	0	0	0	0	0												0	0	0	0	0
0x34	TAMP_MISR	Res.	ITAMP15MF	ITAMP14MF	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value		0	0	0	0	0		0	0	0	0	0	0	0	0	0												0	0	0	0	0
0x3C	TAMP_SCR	Res.	CITAMP15F	CITAMP14F	CITAMP13F	CITAMP12F	CITAMP11F	Res.	CITAMP9F	CITAMP8F	CITAMP7F	CITAMP6F	CITAMP5F	CITAMP4F	CITAMP3F	CITAMP2F	CITAMP1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP5F	CTAMP4F	CTAMP3F	CTAMP2F	CTAMP1F
	Reset value		0	0	0	0	0		0	0	0	0	0	0	0	0	0												0	0	0	0	0
0x40	TAMP_ COUNT1R	COUNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	TAMP_ RPCFGR	RPCFG31	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPCFG4	Res.	Res.	RPCFG1	RPCFG0
	Reset value	1																											0			0	0
0x100 + 0x04*x, (x= 0 to 31)	TAMP_BKPxR	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

44 Inter-integrated circuit interface (I2C)

44.1 I2C introduction

The I2C peripheral handles the interface between the device and the serial I²C (inter-integrated circuit) bus. It provides multicontroller capability, and controls all I²C-bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

The I2C peripheral is also SMBus (system management bus) and PMBus® (power management bus) compatible.

It can use DMA to reduce the CPU load.

44.2 I2C main features

- I²C-bus specification rev03 compatibility:
  - Target and controller modes
  - Multicontroller capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit target addresses (2 addresses, 1 with configurable mask)
  - All 7-bit-addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy-to-use event management
  - Clock stretching (optional)
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters
- SMBus specification rev 3.3 compatibility[(a)]:
  - Hardware PEC (packet error checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.5 standard compatibility
- Independent clock

a. To check the compliance of the GPIOs selected for SMBus with the specified logical levels, refer to the product datasheet.

- Wake-up from Stop mode on address match

For information on I2C instantiation, refer to *Section 44.3: I2C implementation*.

44.3 I2C implementation

This section provides an implementation overview with respect to the I2C instantiation.

Table 499. I2C implementation

I2C features(1)	I2C1	I2C2	I2C3
7-bit addressing mode	X	X	X
10-bit addressing mode	X	X	X
Standard-mode (up to 100 kbit/s)	X	X	X
Fast-mode (up to 400 kbit/s)	X	X	X
Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s)	X	X	X
Independent clock	X	X	X
Wake-up from Stop mode	X	X	X
SMBus/PMBus	X	X	X
SMBus 3.0 VIL/VIH support	X	X	X

1. X = supported.

44.4 I2C functional description

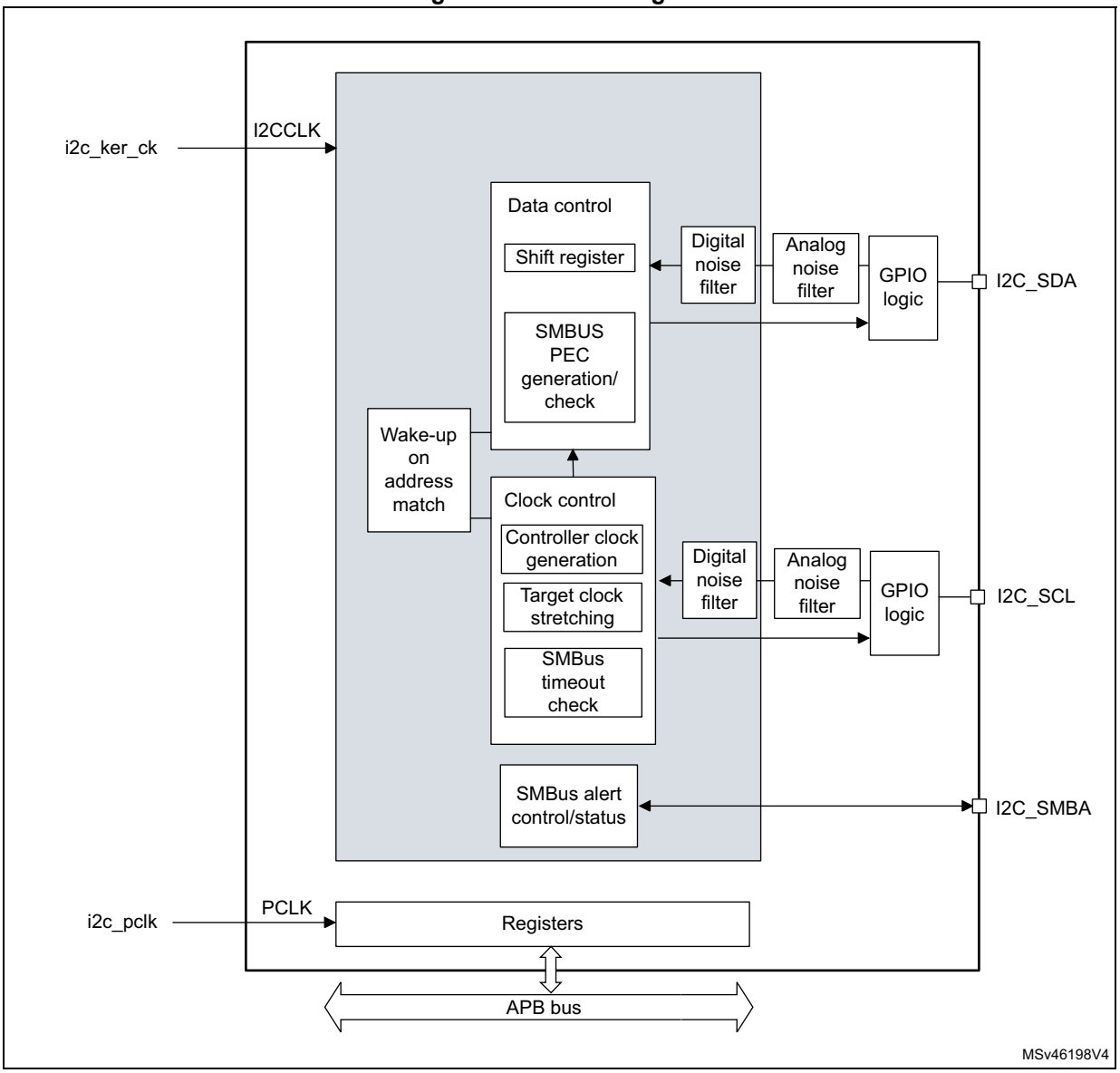
In addition to receiving and transmitting data, the peripheral converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The peripheral is connected to the I²C-bus through a data pin (SDA) and a clock pin (SCL). It supports Standard-mode (up to 100 kHz), Fast-mode (up to 400 kHz), and Fast-mode Plus (up to 1 MHz) I²C-bus.

The peripheral can also be connected to an SMBus, through the data pin (SDA), the clock pin (SCL), and an optional SMBus alert pin (SMBA).

The independent clock function allows the I2C communication speed to be independent of the i2c_pclk frequency.

44.4.1 I2C block diagram

Figure 866. Block diagram



44.4.2 I2C pins and internal signals

Table 500. I2C input/output pins

Pin name	Signal type	Description
I2C_SDA	Bidirectional	I²C-bus data
I2C_SCL	Bidirectional	I²C-bus clock
I2C_SMBA	Bidirectional	SMBus alert

Table 501. I2C internal input/output signals

Internal signal name	Signal type	Description
i2c_ker_ck	Input	I2C kernel clock, also named I2CCLK in this document
i2c_pclk	Input	I2C APB clock
i2c_it	Output	I2C interrupts, refer to *Table 515* for the list of interrupt sources
i2c_rx_dma	Output	I2C receive data DMA request (I2C_RX)
i2c_tx_dma	Output	I2C transmit data DMA request (I2C_TX)

44.4.3 I2C clock requirements

The I2C kernel is clocked by i2c_ker_ck.

The i2c_ker_ck period t_{I2CCLK} must respect the following conditions:

$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$

$t_{I2CCLK} < t_{HIGH}$

where t_{LOW} is the SCL low time, t_{HIGH} is the SCL high time, and $t_{filters}$ is the sum of the analog and digital filter delays (when enabled).

The digital filter delay is DNF[3:0] x t_{I2CCLK}.

The i2c_pclk clock period t_{PCLK} must respect the condition $t_{PCLK} < 4/3\ t_{SCL}$, where t_{SCL} is the SCL period.

Caution: When the I2C kernel is clocked by i2c_pclk, this clock must respect the conditions for t_{I2CCLK}.

44.4.4 I2C mode selection

The peripheral can operate as:

- Target transmitter
- Target receiver
- Controller transmitter
- Controller receiver

By default, the peripheral operates in target mode. It automatically switches from target to controller mode upon generating START condition, and from controller to target mode upon arbitration loss or upon generating STOP condition. This allows the use of the I2C peripheral in a multicontroller I²C-bus environment.

Communication flow

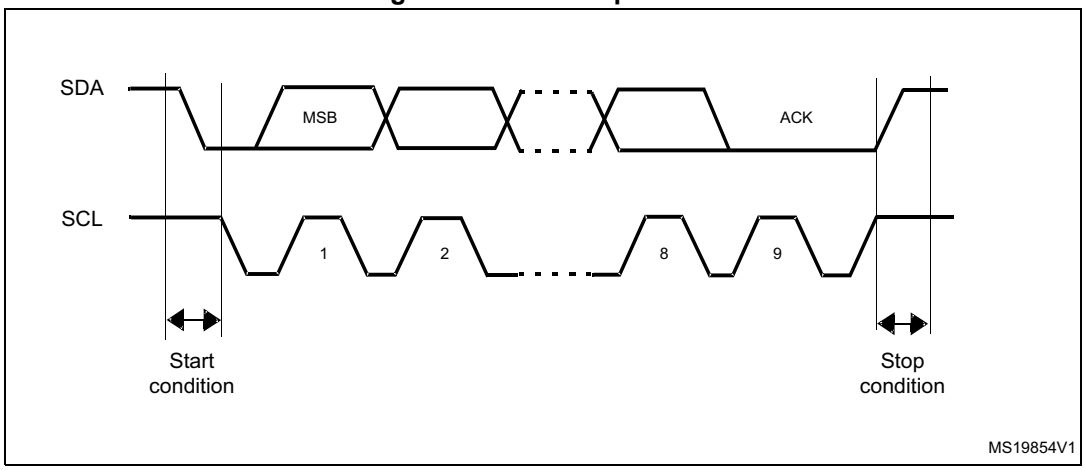
In controller mode, the I2C peripheral initiates a data transfer and generates the clock signal. Serial data transfers always begin with a START condition and end with a STOP condition. Both START and STOP conditions are generated in controller mode by software.

In target mode, the peripheral recognizes its own 7-bit or 10-bit address, and the general call address. The general call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The address is contained in the first byte (7-bit addressing) or in the first two bytes (10-bit addressing) following the START condition. The address is always transmitted in controller mode.

The following figure shows the transmission of a single byte. The controller generates nine SCL pulses. The transmitter sends the eight data bits to the receiver with the SCL pulses 1 to 8. Then the receiver sends the acknowledge bit to the transmitter with the ninth SCL pulse.

Figure 867. I²C-bus protocol



The acknowledge can be enabled or disabled by software. The own addresses of the I2C peripheral can be selected by software.

44.4.5 I2C initialization

Enabling and disabling the peripheral

Before enabling the I2C peripheral, configure and enable its clock through the RCC, and initialize its control registers.

The I2C peripheral can then be enabled by setting the PE bit of the I2C_CR1 register.

Disabling the I2C peripheral by clearing the PE bit resets the I2C peripheral. Refer to *Section 44.4.6* for more details.

Noise filters

Before enabling the I2C peripheral by setting the PE bit of the I2C_CR1 register, the user must configure the analog and/or digital noise filters, as required.

The analog noise filter on the SDA and SCL inputs complies with the I²C-bus specification which requires, in Fast-mode and Fast-mode Plus, the suppression of spikes shorter than 50 ns. Enabled by default, it can be disabled by setting the ANFOFF bit.

The digital filter is controlled through the DNF[3:0] bitfield of the I2C_CR1 register. When it is enabled, the internal SCL and SDA signals only take the level of their corresponding I²C-bus line when remaining stable for more than DNF[3:0] periods of i2c_ker_ck. This allows suppressing spikes shorter than the filtering capacity period programmable from one to fifteen i2c_ker_ck periods.

The following table compares the two filters.

Table 502. Comparison of analog and digital filters

Item	Analog filter	Digital filter
Filtering capacity(1)	≥ 50 ns	One to fifteen i2c_ker_ck periods
Benefits	Available in Stop mode	– Programmable filtering capacity – Extra filtering capability versus I²C-bus specification requirements – Stable filtering capacity
Drawbacks	Filtering capacity variation with temperature, voltage, and silicon process	Wake-up from Stop mode on address match not supported when the digital filter is enabled

1. Maximum duration of spikes that the filter can suppress

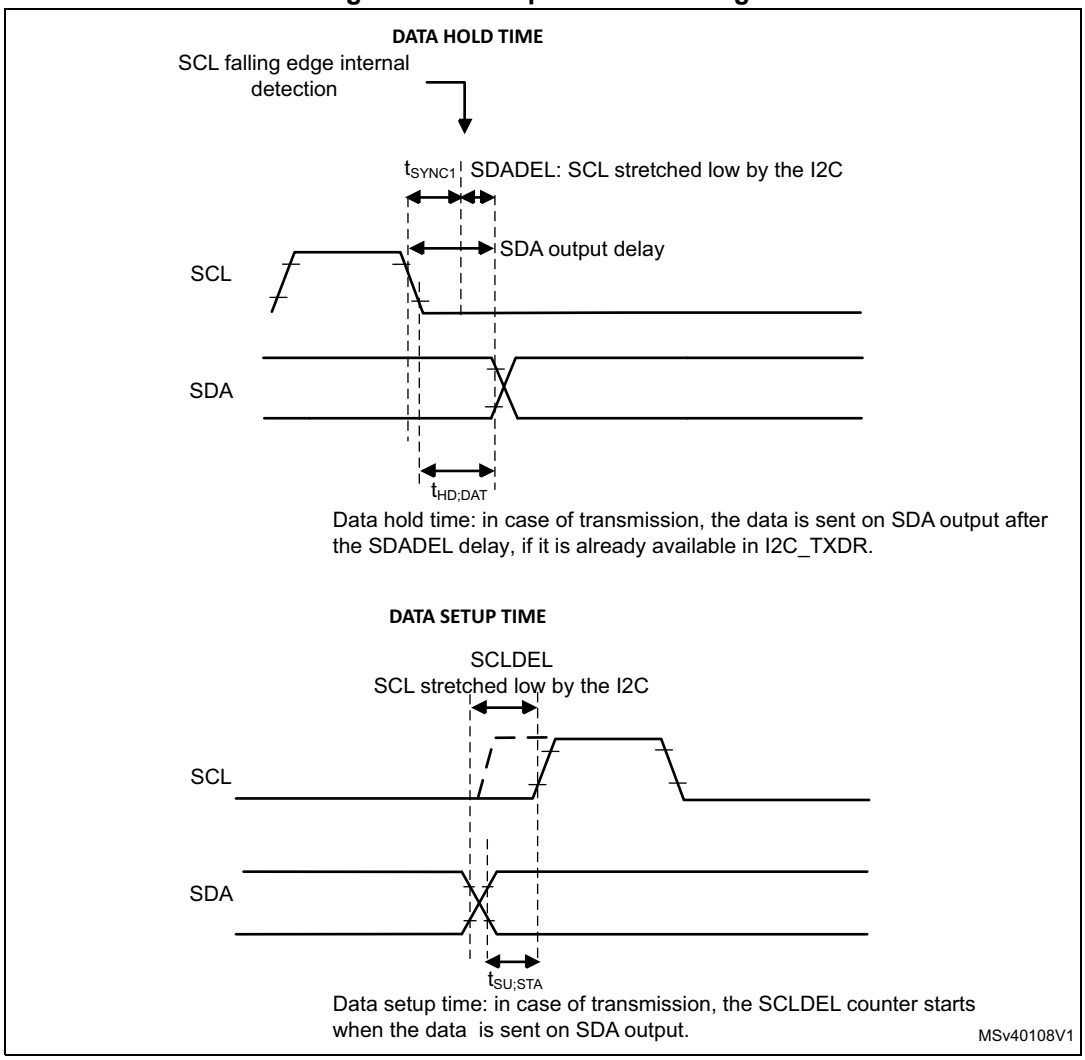
Caution: The filter configuration cannot be changed when the I2C peripheral is enabled.

I2C timings

To ensure correct data hold and setup times, the corresponding timings must be configured through the PRESC[3:0], SCLDEL[3:0], and SDADEL[3:0] bitfields of the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C configuration* window.

Figure 868. Setup and hold timings



When the SCL falling edge is internally detected, the delay t_{SDADEL} (impacting the hold time $t_{HD;DAT}$) is inserted before sending SDA output:

t_{SDADEL} = SDADEL x t_{PRESC} + t_{I2CCLK}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}.

The total SDA output delay is:

t_{SYNC1} + {[SDADEL x (PRESC + 1) + 1] x t_{I2CCLK}}

The t_{SYNC1} duration depends upon:

- SCL falling slope
- input delay $t_{AF(min)} < t_{AF} < t_{AF(max)}$ introduced by the analog filter (if enabled)
- input delay t_{DNF} = DNF x t_{I2CCLK} introduced by the digital filter (if enabled)
- delay due to SCL synchronization to i2c_ker_ck clock (two to three i2c_ker_ck periods)

To bridge the undefined region of the SCL falling edge, the user must set SDADEL[3:0] so as to fulfill the following condition:

$\{t_{f(max)} + t_{HD;DAT(min)} - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\} \leq$ SDADEL

SDADEL $\leq \{t_{HD;DAT\,(max)} - t_{AF(max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$

Note: $t_{AF(min)}$ and $t_{AF(max)}$ are only part of the condition when the analog filter is enabled. Refer to the device datasheet for t_{AF} values.

The $t_{HD;DAT}$ time can at maximum be 3.45 µs for Standard-mode, 0.9 µs for Fast-mode, and 0.45 µs for Fast-mode Plus. It must be lower than the maximum of $t_{VD;DAT}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. When it stretches SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case. The previous condition then becomes:

$$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - t_{AF\ (max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$$

Note: This condition can be violated when NOSTRETCH = 0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL[3:0] value.

After t_{SDADEL}, or after sending SDA output when the target had to stretch the clock because the data was not yet written in I2C_TXDR register, the SCL line is kept at low level during the setup time. This setup time is $t_{SCLDEL} = (SCLDEL + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$. t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

To bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL[3:0] so as to fulfill the following condition:

$$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC + 1) \times t_{I2CCLK}]\} - 1 \leq SCLDEL$$

Refer to the following table for t_f, t_r, $t_{HD;DAT}$, $t_{VD;DAT}$, and $t_{SU;DAT}$ standard values.

Use the SDA and SCL real transition time values measured in the application to widen the scope of allowed SDADEL[3:0] and SCLDEL[3:0] values. Use the maximum SDA and SCL transition time values defined in the standard to make the device work reliably regardless of the application.

Note: At every clock pulse, after SCL falling edge detection, I2C operating as controller or target stretches SCL low during at least [(SDADEL + SCLDEL + 1) x (PRESC + 1) + 1] x t_{I2CCLK}, in both transmission and reception modes. In transmission mode, if the data is not yet written in I2C_TXDR when SDA delay elapses, the I2C peripheral keeps stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

When the NOSTRETCH bit is set in target mode, the SCL is not stretched. The SDADEL[3:0] must then be programmed so that it ensures a sufficient setup time.

Table 503. I²C-bus and SMBus specification data setup and hold times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	µs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Additionally, in controller mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0], and SCLL[7:0] bitfields of the I2C_TIMINGR register.

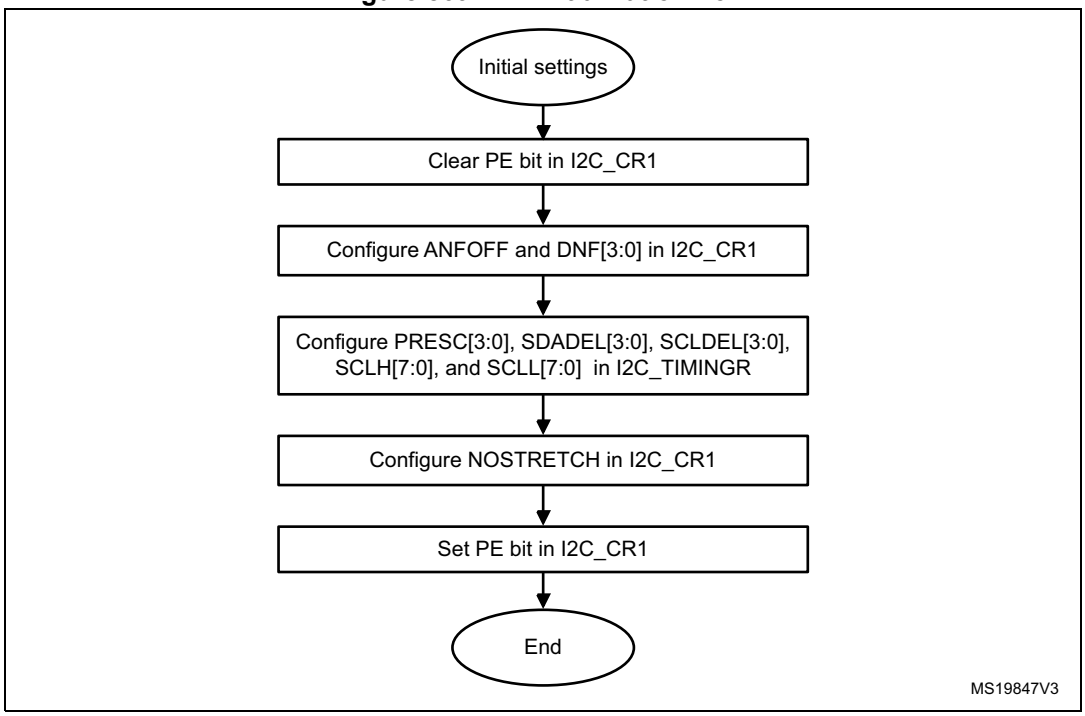
When the SCL falling edge is internally detected, the I2C peripheral releasing the SCL output after the delay t_{SCLL} = (SCLL + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLL} delay impacts the SCL low time t_{LOW}.

When the SCL rising edge is internally detected, the I2C peripheral forces the SCL output to low level after the delay t_{SCLH} = (SCLH + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLH} impacts the SCL high time t_{HIGH}.

Refer to *I2C controller initialization* for more details.

Caution: Changing the timing configuration and the NOSTRETCH configuration is not allowed when the I2C peripheral is enabled. Like the timing settings, the target NOSTRETCH settings must also be done before enabling the peripheral. Refer to *I2C target initialization* for more details.

Figure 869. I2C initialization flow



44.4.6 I2C reset

The reset of the I2C peripheral is performed by clearing the PE bit of the I2C_CR1 register. It has the effect of releasing the SCL and SDA lines. Internal state machines are reset and the communication control bits and the status bits revert to their reset values. This reset does not impact the configuration registers.

The impacted register bits are:

1. I2C_CR2 register: START, STOP, PECBYTE, and NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, PECERR, TIMEOUT, ALERT, and OVR

PE must be kept low during at least three APB clock cycles to perform the I2C reset. To ensure this, perform the following software sequence:

1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

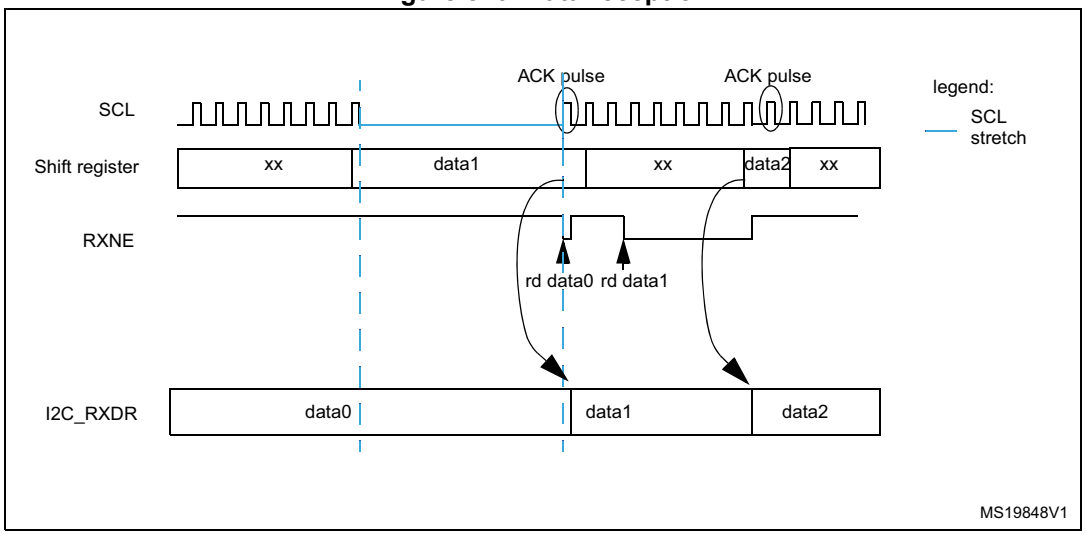
44.4.7 I2C data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the eighth SCL pulse (when the complete data byte is received), the shift register is copied into the I2C_RXDR register if it is empty (RXNE = 0). If RXNE = 1, which means that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch occurs between the eighth and the ninth SCL pulse (before the acknowledge pulse).

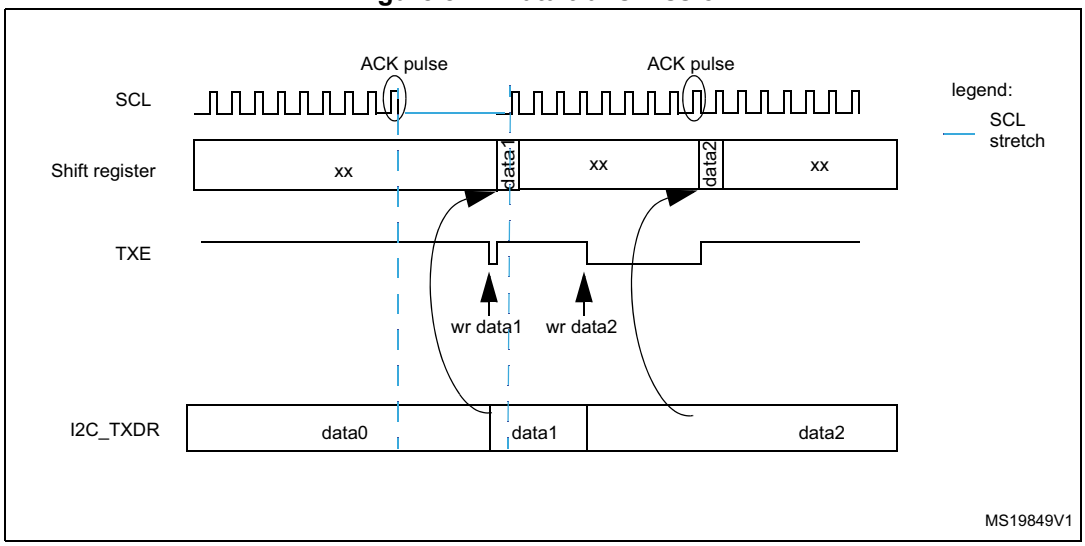
Figure 870. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE = 0), its content is copied into the shift register after the ninth SCL pulse (the acknowledge pulse). Then the shift register content is shifted out on the SDA line. If TXE = 1, which means that no data is written yet in I2C_TXDR, the SCL line is stretched low until I2C_TXDR is written. The stretch starts after the ninth SCL pulse.

Figure 871. Data transmission



Hardware transfer management

The I2C features an embedded byte counter to manage byte transfer and to close the communication in various modes, such as:

- NACK, STOP and ReSTART generation in controller mode
- ACK control in target receiver mode
- PEC generation/checking

In controller mode, the byte counter is always used. By default, it is disabled in target mode. It can be enabled by software, by setting the SBC (target byte control) bit of the I2C_CR1 register.

The number of bytes to transfer is programmed in the NBYTES[7:0] bitfield of the I2C_CR2 register. If this number is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected, by setting the RELOAD bit of the I2C_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES[7:0] is transferred (when the associated counter reaches zero), and an interrupt is generated if TCIE is set. SCL is stretched as long as the TCR flag is set. TCR is cleared by software when NBYTES[7:0] is written to a non-zero value.

When NBYTES[7:0] is reloaded with the last number of bytes to transfer, the RELOAD bit must be cleared.

When RELOAD = 0 in controller mode, the counter can be used in two modes:

- **Automatic end** (AUTOEND = 1 in the I2C_CR2 register). In this mode, the controller automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred.
- **Software end** (AUTOEND = 0 in the I2C_CR2 register). In this mode, a software action is expected once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit of the I2C_CR2 register is set. This mode must be used when the controller wants to send a RESTART condition.

Caution: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 504. I2C configuration

Function	SBC bit	RELOAD bit	AUTOEND bit
Controller Tx/Rx NBYTES + STOP	X	0	1
Controller Tx/Rx + NBYTES + RESTART	X	0	0
Target Tx/Rx, all received bytes ACKed	0	X	X
Target Rx with ACK control	1	1	X

44.4.8 I2C target mode

I2C target initialization

To work in target mode, the user must enable at least one target address. The I2C_OAR1 and I2C_OAR2 registers are available to program the target own addresses OA1 and OA2, respectively.

OA1 can be configured either in 7-bit (default) or in 10-bit addressing mode, by setting the OA1MODE bit of the I2C_OAR1 register.

OA1 is enabled by setting the OA1EN bit of the I2C_OAR1 register.

If an additional target addresses are required, the second target address OA2 can be configured. Up to seven OA2 LSBs can be masked, by configuring the OA2MSK[2:0] bitfield of the I2C_OAR2 register. Therefore, for OA2MSK[2:0] configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6], or OA2[7] are compared with the received address. When OA2MSK[2:0] is other than 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX) and they are not acknowledged. If OA2MSK[2:0] = 7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

When enabled through the specific bit, the reserved addresses can be acknowledged if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK[2:0] = 0.

OA2 is enabled by setting the OA2EN bit of the I2C_OAR2 register.

The general call address is enabled by setting the GCEN bit of the I2C_CR1 register.

When the I2C peripheral is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the target uses its clock stretching capability, which means that it stretches the SCL signal at low level when required, to perform software actions. If the controller does not

support clock stretching, I2C must be configured with NOSTRETCH = 1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled, the user must read the ADDCODE[6:0] bitfield of the I2C_ISR register to check which address matched. The DIR flag must also be checked to know the transfer direction.

Target with clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is zero (default), the I2C peripheral operating as an I²C-bus target stretches the SCL signal in the following situations:

- The ADDR flag is set and the received address matches with one of the enabled target addresses.
  The stretch is released when the software clears the ADDR flag by setting the ADDRCF bit.
- In transmission, the previous data transmission is completed and no new data is written in I2C_TXDR register, or the first data byte is not written when the ADDR flag is cleared (TXE = 1).
  The stretch is released when the data is written to the I2C_TXDR register.
- In reception, the I2C_RXDR register is not read yet and a new data reception is completed.
  The stretch is released when I2C_RXDR is read.
- In target byte control mode (SBC bit set) with reload (RELOAD bit set), the last data byte transfer is finished (TCR bit set).
  The stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] bitfield.
- After SCL falling edge detection.
  The stretch is released after [(SDADEL + SCLDEL + 1) x (PRESC+ 1) + 1] x t_{I2CCLK} period.

Target without clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is set, the I2C peripheral operating as an I²C-bus target does not stretch the SCL signal.

The SCL clock is not stretched while the ADDR flag is set.

In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, it ensures that the OVR status is provided, even for the first data to be transmitted.

In reception, the data must be read from the I2C_RXDR register before the ninth SCL pulse (ACK pulse) of the next data byte occurs. If not, an overrun occurs, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Target byte control mode

To allow byte ACK control in target reception mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. This is required to comply with SMBus standards.

The reload mode must be selected to allow byte ACK control in target reception mode (RELOAD = 1). To get control of each byte, NBYTES[7:0] must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the eighth and the ninth SCL pulse. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit of the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and the next byte can be received.

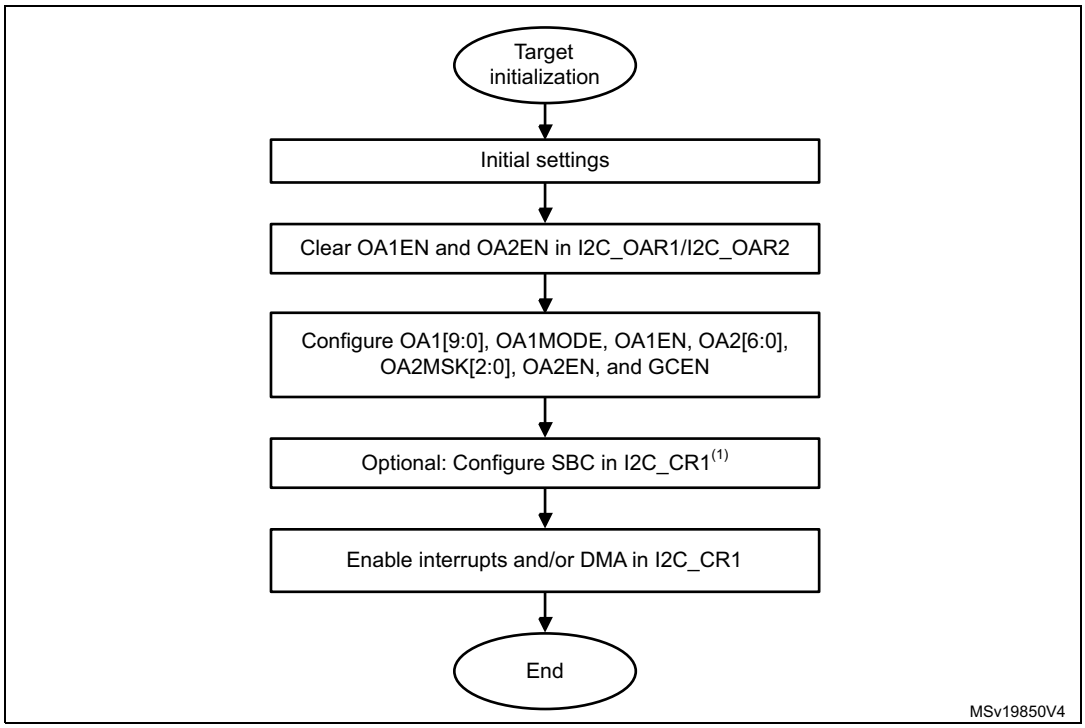
NBYTES[7:0] can be loaded with a value greater than 0x1. Receiving then continues until the corresponding number of bytes are received.

Note: The SBC bit must be configured when the I2C peripheral is disabled, when the target is not addressed, or when ADDR = 1.

The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.

Caution: The target byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

Figure 872. Target initialization flow



1. SBC must be set to support SMBus features.

Target transmitter

A transmit interrupt status (TXIS) flag is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit of the I2C_CR1 register is set.

The TXIS flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

When NACK is received, the NACKF flag is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit of the I2C_CR1 register is set. The target automatically releases the SCL and SDA lines to let the controller perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When STOP is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag of the I2C_ISR register is set and an interrupt is generated. In most applications, the SBC bit is usually programmed to 0. In this case, if TXE = 0 when the target address is received (ADDR = 1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register, by setting the TXE bit in order to program a new data byte.

In target byte control mode (SBC = 1), the number of bytes to transmit must be programmed in NBYTES[7:0] in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0].

Caution: When NOSTRETCH = 1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine to program the first data byte. The first data byte to send must be previously programmed in the I2C_TXDR register:

- This data can be the one written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to send, the I2C_TXDR register can be flushed, by setting the TXE bit, to program a new data byte. The STOPF bit must be cleared only after these actions. This guarantees that they are executed before the first data transmission starts, following the address acknowledge.

  If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

  If a TXIS event (transmit interrupt or transmit DMA request) is required, the user must set the TXIS bit in addition to the TXE bit, to generate the event.

Figure 873. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0

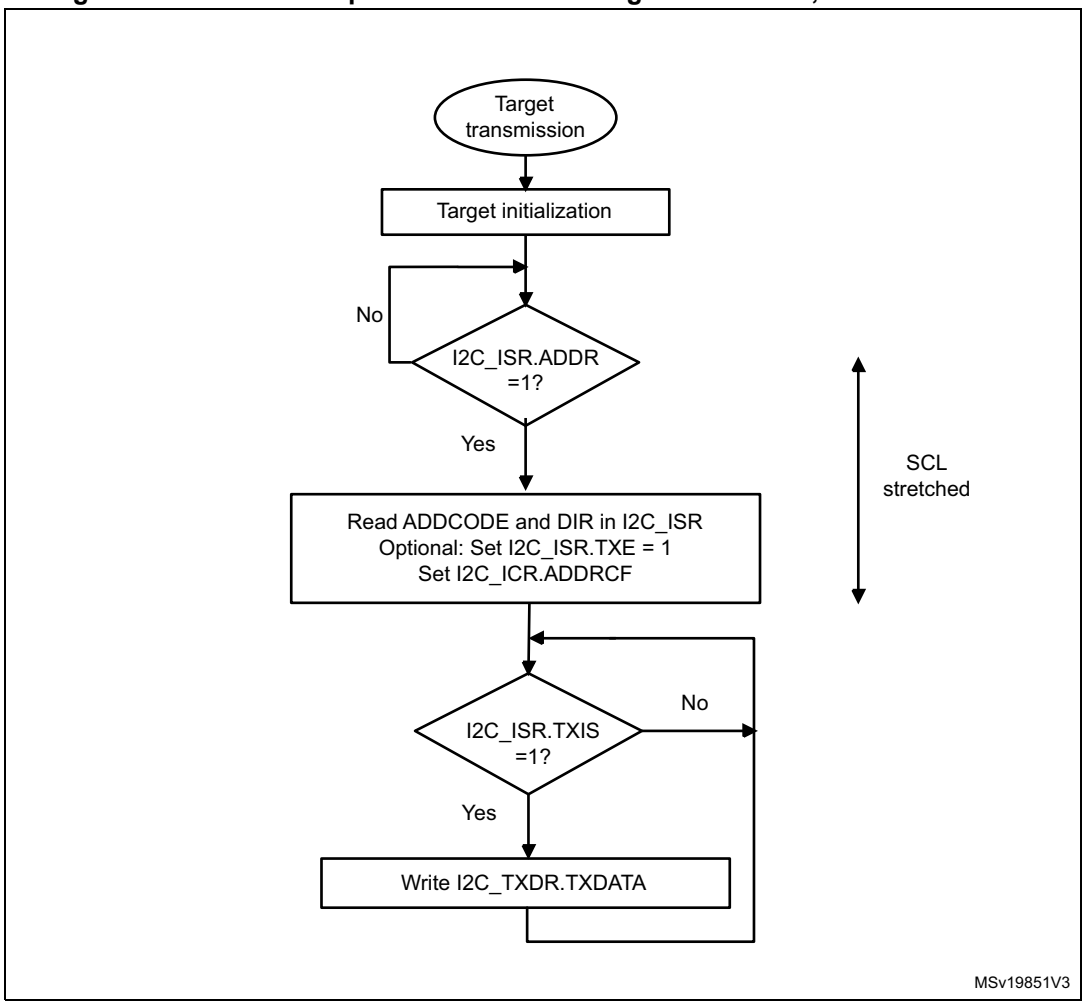


Figure 874. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1

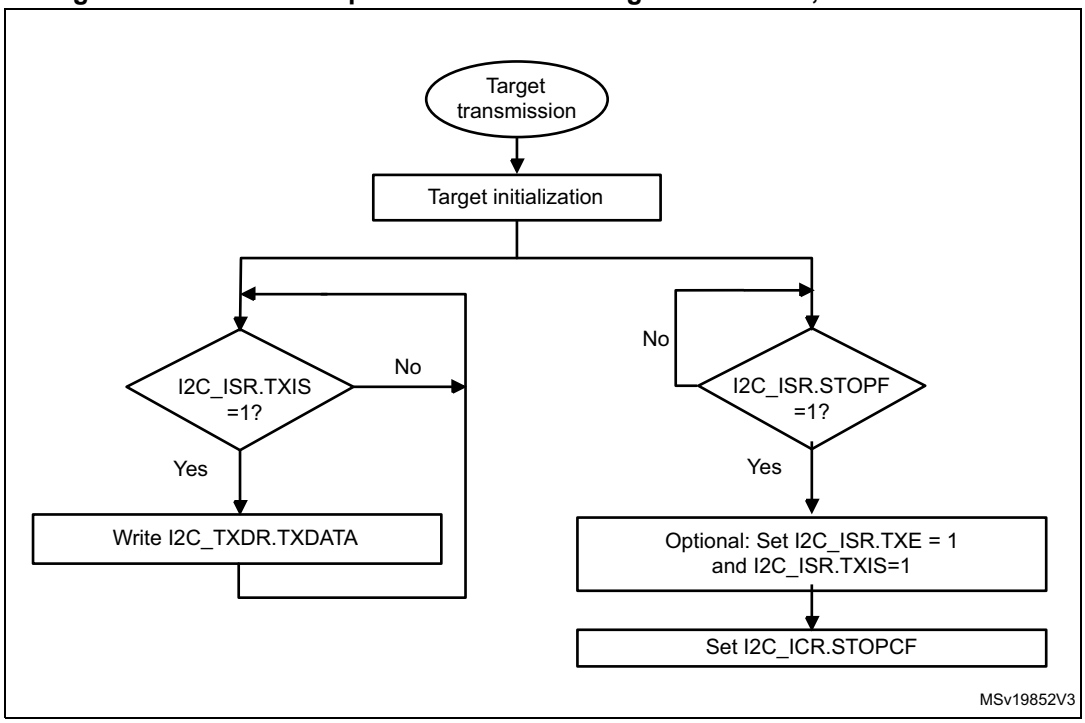
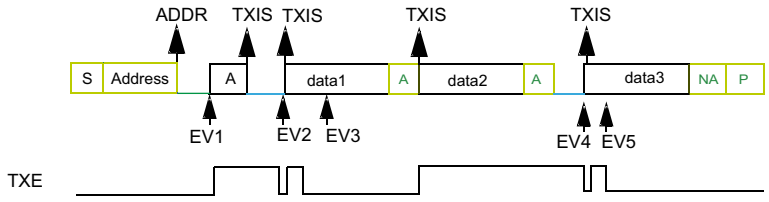


Figure 875. Transfer bus diagrams for I2C target transmitter (mandatory events only)

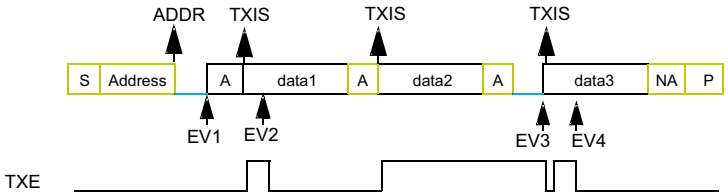
Example I2C target transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



legend:
transmission
reception
SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

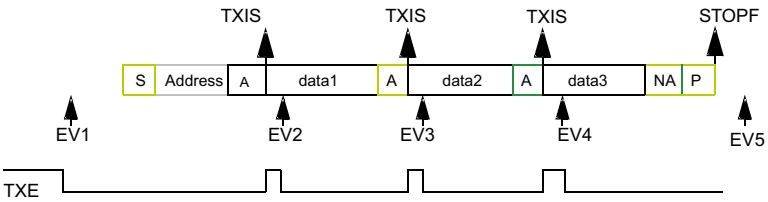
Example I2C target transmitter 3 bytes without 1st data flush, NOSTRETCH=0:



legend :
transmission
reception
SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)

Example I2C target transmitter 3 bytes, NOSTRETCH=1:



legend:
transmission
reception
SCL stretch

EV1: wr data1
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)
EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

MSv19853V3

Target receiver

The RXNE bit of the I2C_ISR register is set when the I2C_RXDR is full, which generates an interrupt if the RXIE bit of the I2C_CR1 register is set. RXNE is cleared when I2C_RXDR is read.

When STOP condition is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag in the I2C_ISR register is set and an interrupt is generated.

Figure 876. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0

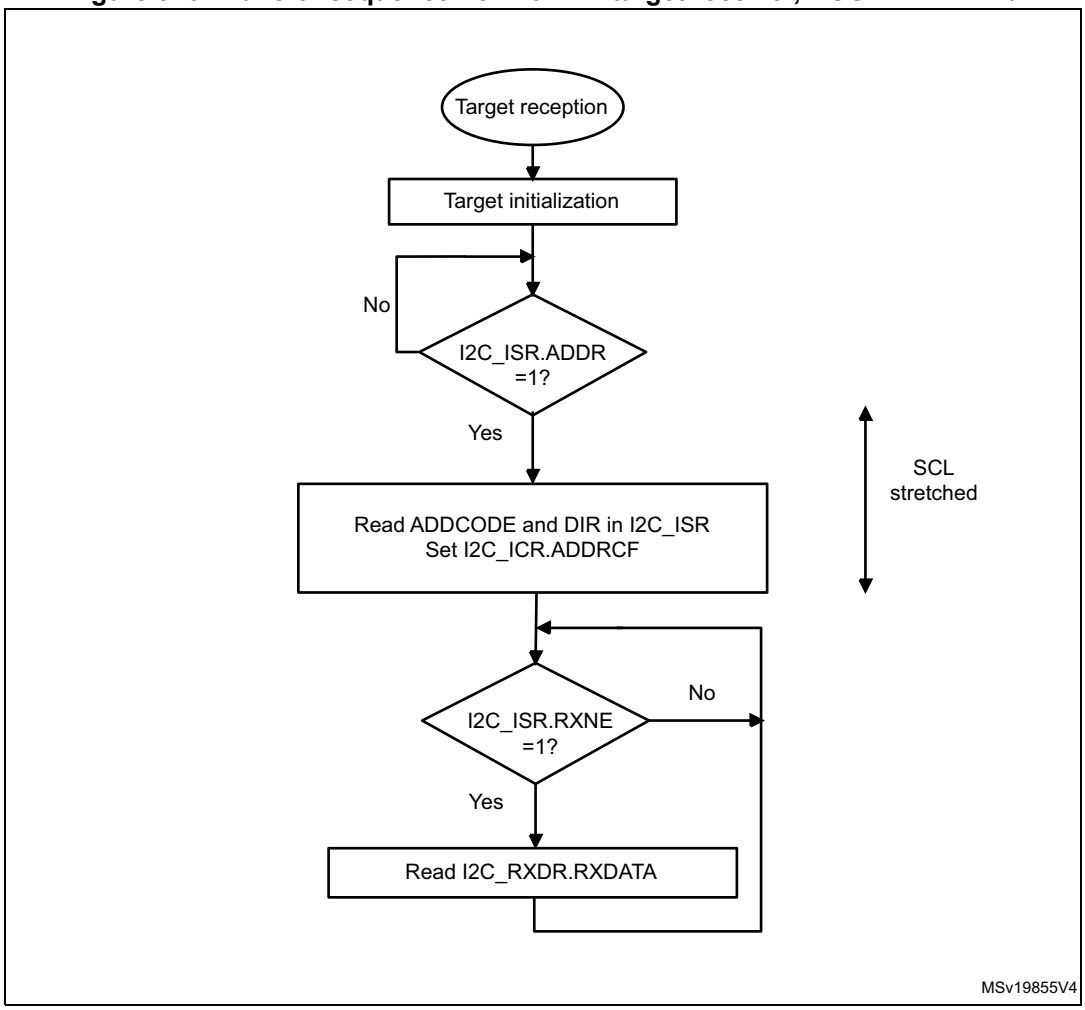


Figure 877. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1

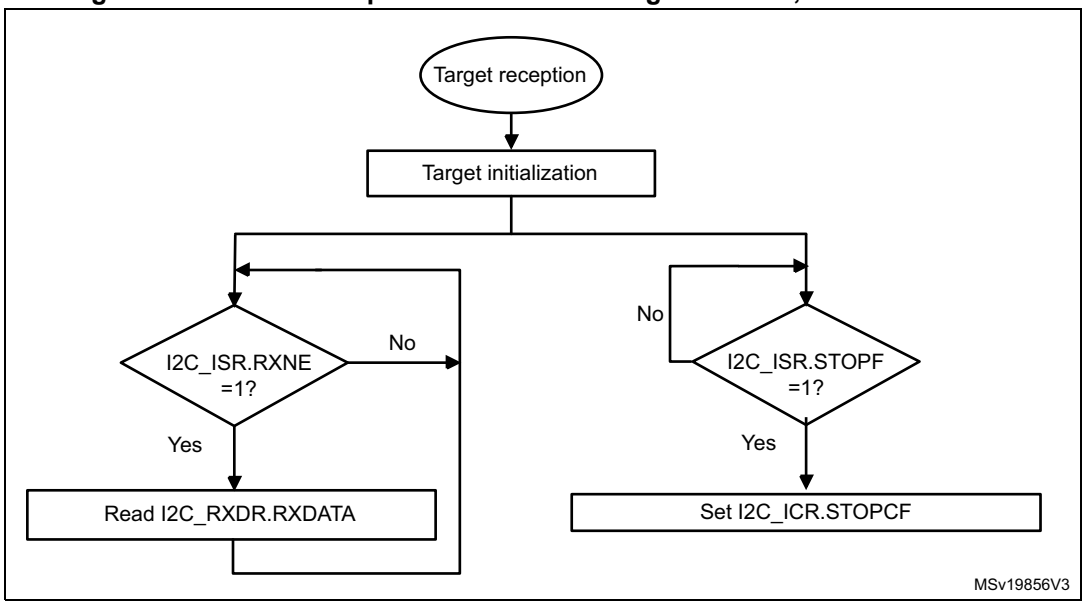
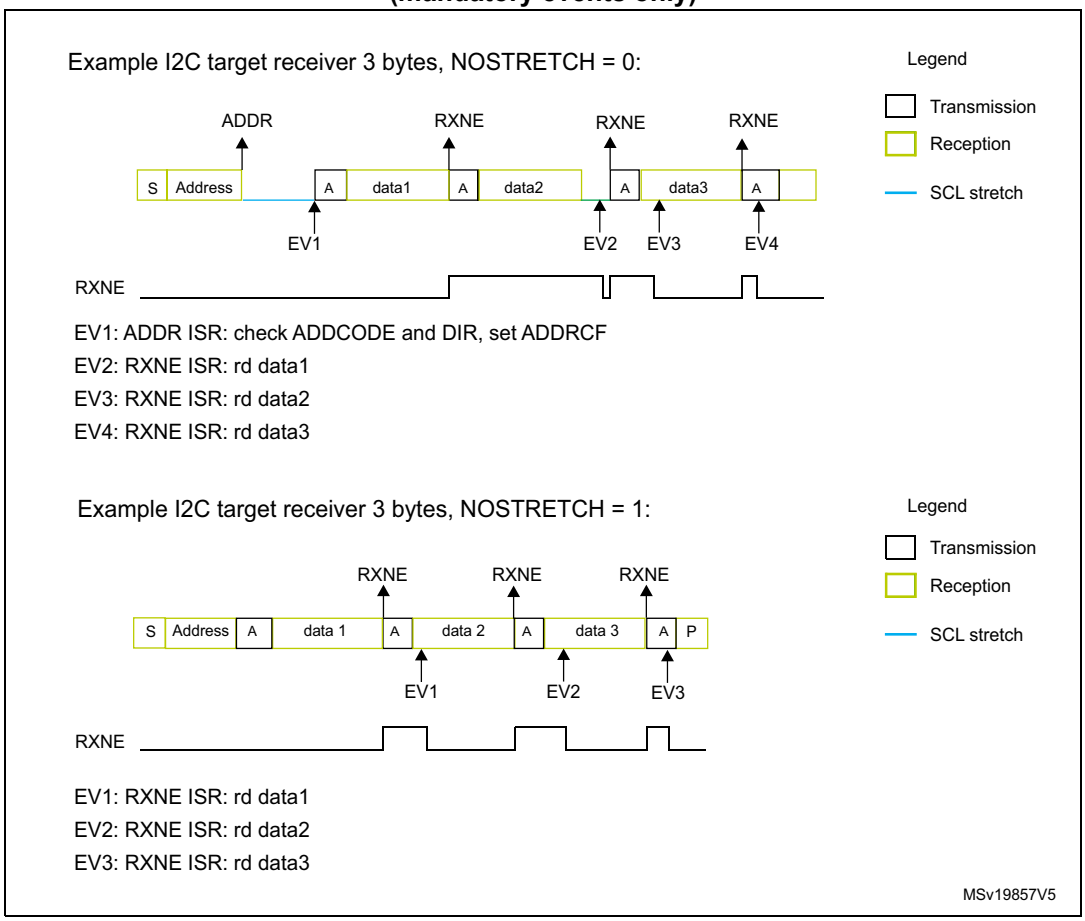


Figure 878. Transfer bus diagrams for I2C target receiver (mandatory events only)



44.4.9 I2C controller mode

I2C controller initialization

Before enabling the peripheral, the I2C controller clock must be configured, by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C Configuration* window.

A clock synchronization mechanism is implemented in order to support multicontroller environment and target clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bitfield of the I2C_TIMINGR register.

I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C ties SCL to low level once the SCLH counter reaches the value programmed in the SCLH[7:0] bitfield of the I2C_TIMINGR register.

Consequently the controller clock period is:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+ 1) + (SCLL+ 1)] \times (PRESC+ 1) \times t_{I2CCLK}\}$

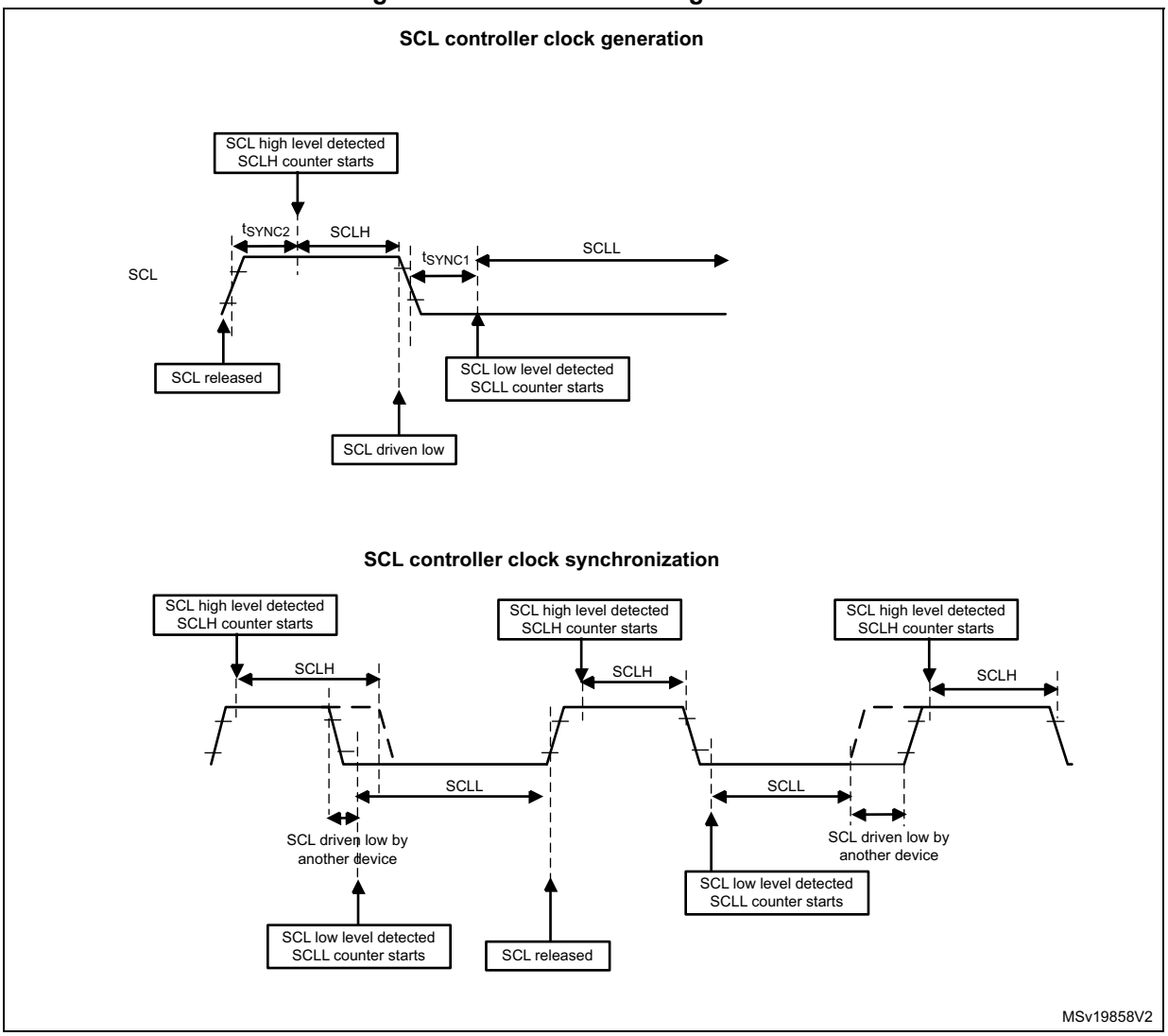
The duration of t_{SYNC1} depends upon:

- SCL falling slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

The duration of t_{SYNC2} depends upon:

- SCL rising slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

Figure 879. Controller clock generation



Caution: For compliance with the I²C-bus or SMBus specification, the controller clock must respect the timings in the following table.

Table 505. I²C-bus and SMBus specification clock timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency	-	100	-	400	-	1000	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	0.26	-	4.7	-	
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	
t_{LOW}	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	
t_{HIGH}	High period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Note: The SCLL[7:0] bitfield also determines the t_{BUF} and $t_{SU:STA}$ timings and SCLH[7:0] the $t_{HD:STA}$ and $t_{SU:STO}$ timings.

Refer to *Section 44.4.10* for examples of I2C_TIMINGR settings versus the i2c_ker_ck frequency.

Controller communication initialization (address phase)

To initiate the communication with a target to address, set the following bitfields of the I2C_CR2 register:

- ADD10: addressing mode (7-bit or 10-bit)
- SADD[9:0]: target address to send
- RD_WRN: transfer direction
- HEAD10R: in case of 10-bit address read, this bit determines whether the header only (for direction change) or the complete address sequence is sent.
- NBYTES[7:0]: the number of bytes to transfer; if equal to or greater than 255 bytes, the bitfield must initially be set to 0xFF.

Note: Changing these bitfields is not allowed as long as the START bit is set.

Before launching the communication, make sure that the I²C-bus is idle. This can be checked using the bus idle detection function or by verifying that the IDR bits of the GPIOs selected as SDA and SCL are set. Any low-level incident on the I²C-bus lines that coincides with the START condition asserted by the I2C peripheral may cause its deadlock if not filtered out by the input filters. If such incidents cannot be prevented, design the software so that it restores the normal operation of the I2C peripheral in case of a deadlock, by toggling the PE bit of the I2C_CR1 register.

To launch the communication, set the START bit of the I2C_CR2 register. The controller then automatically sends a START condition followed by the target address, either immediately if the BUSY flag is low, or t_{BUF} time after the BUSY flag transits from high to low state. The BUSY flag is set upon sending the START condition.

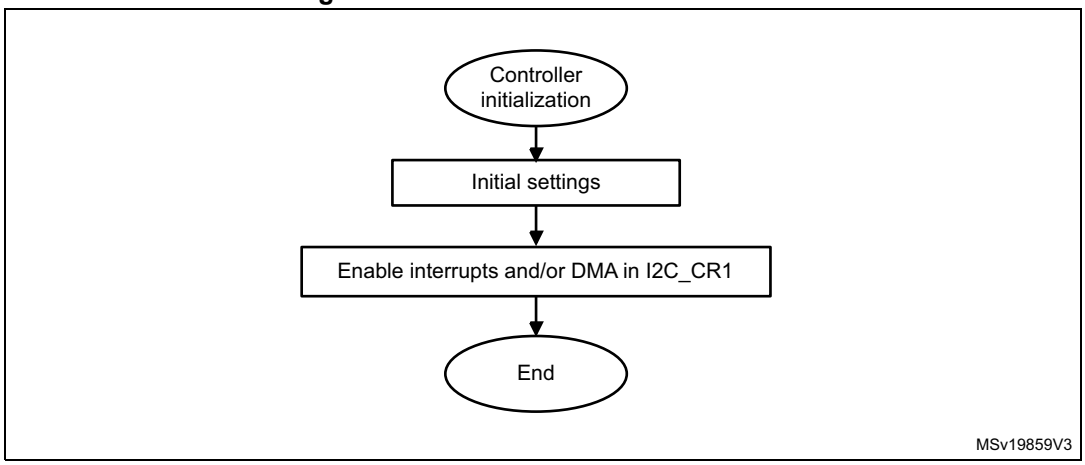
In case of an arbitration loss, the controller automatically switches back to target mode and can acknowledge its own address if it is addressed as a target.

Note: The START bit is reset by hardware when the target address is sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware upon arbitration loss.

In 10-bit addressing mode, the controller automatically keeps resending the target address in a loop until the first address byte (first seven address bits) is acknowledged by the target. Setting the ADDRCF bit makes I2C quit that loop.
If the I2C peripheral is addressed as a target (ADDR = 1) while the START bit is set, the I2C peripheral switches to target mode and the START bit is cleared.

Note: The same procedure is applied for a repeated START condition. In this case, BUSY = 1.

Figure 880. Controller initialization flow

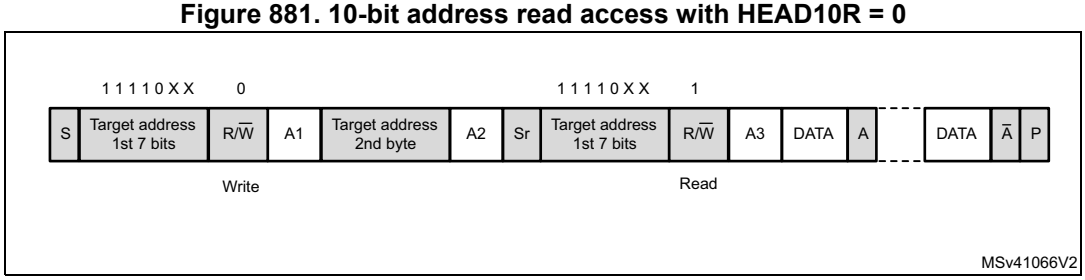


Initialization of a controller receiver addressing a 10-bit address target

If the target address is in 10-bit format, the user can choose to send the complete read sequence, by clearing the HEAD10R bit of the I2C_CR2 register. In this case, the controller automatically sends the following complete sequence after the START bit is set:

(RE)START + Target address 10-bit header Write + Target address second byte + (RE)START + Target address 10-bit header Read.

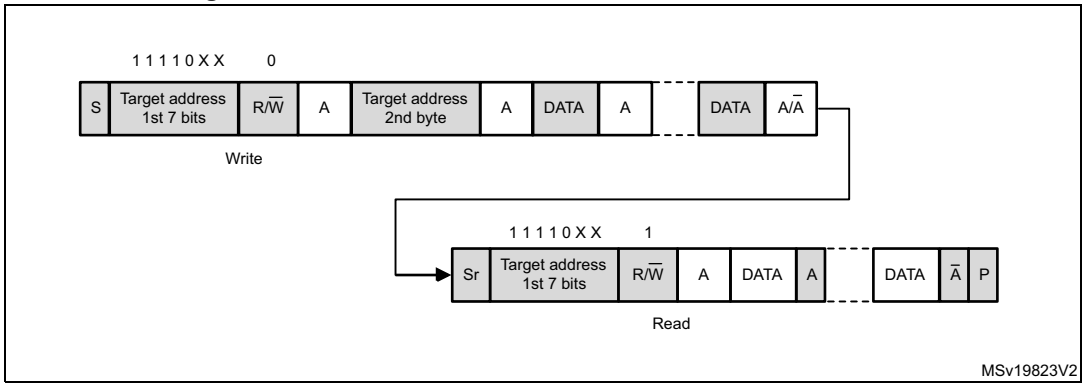
Figure 881. 10-bit address read access with HEAD10R = 0



If the controller addresses a 10-bit address target, transmits data to this target and then reads data from the same target, a controller transmission flow must be done first. Then a repeated START is set with the 10-bit target address configured with HEAD10R = 1. In this case, the controller sends this sequence:

RESTART + Target address 10-bit header Read.

Figure 882. 10-bit address read access with HEAD10R = 1



Controller transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the ninth SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit of the I2C_CR1 register is set. The flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to transmit is greater than 255, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and the number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a STOP condition is automatically sent.
- In software end mode (AUTOEND = 0), the TC flag is set and the SCL line is stretched low, to perform software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

When a NACK is received, the TXIS flag is not set and a STOP condition is automatically sent. The NACKF flag of the I2C_ISR register is set. An interrupt is generated if the NACKIE bit is set.

Figure 883. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes

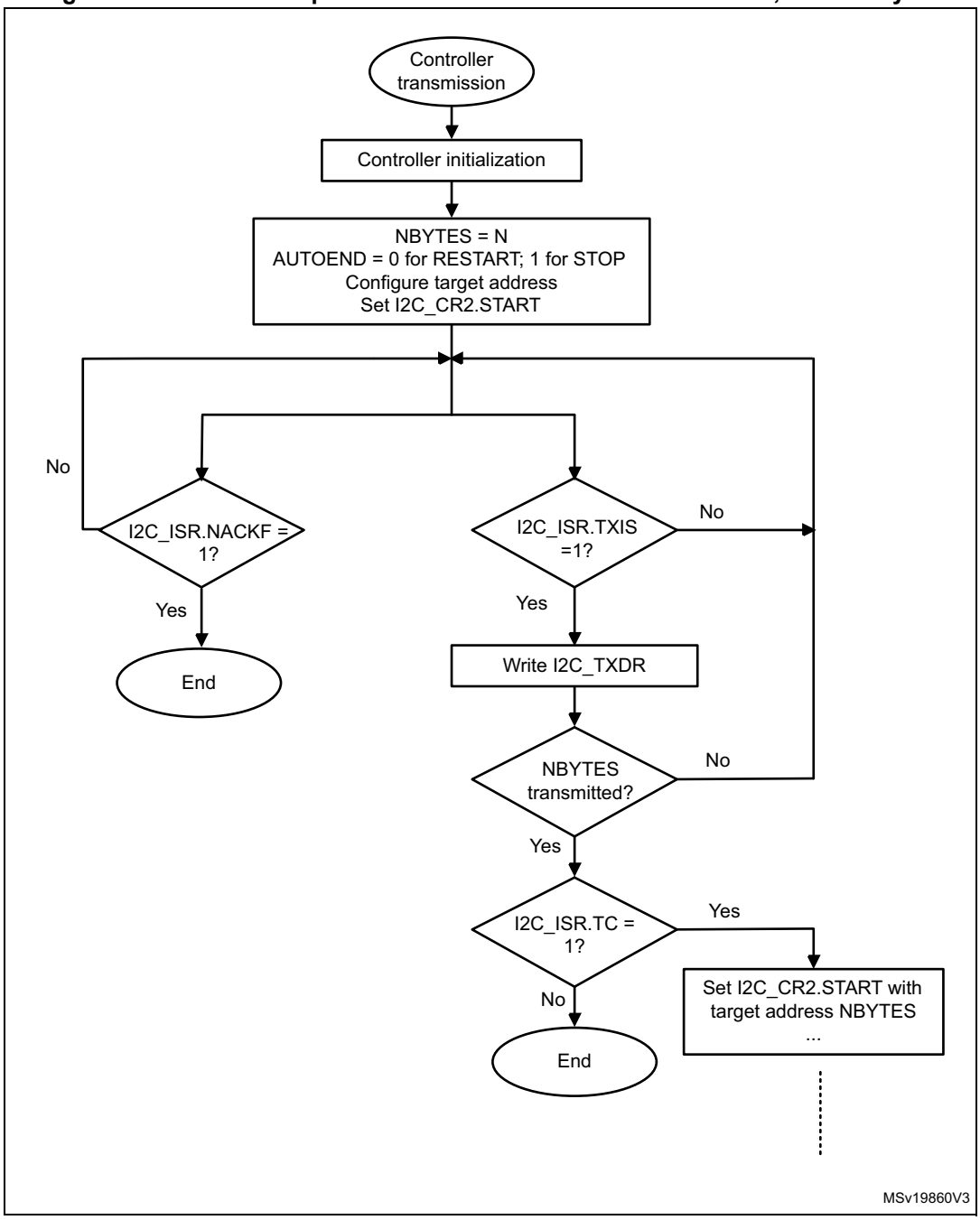


Figure 884. Transfer sequence flow for I2C controller transmitter, N > 255 bytes

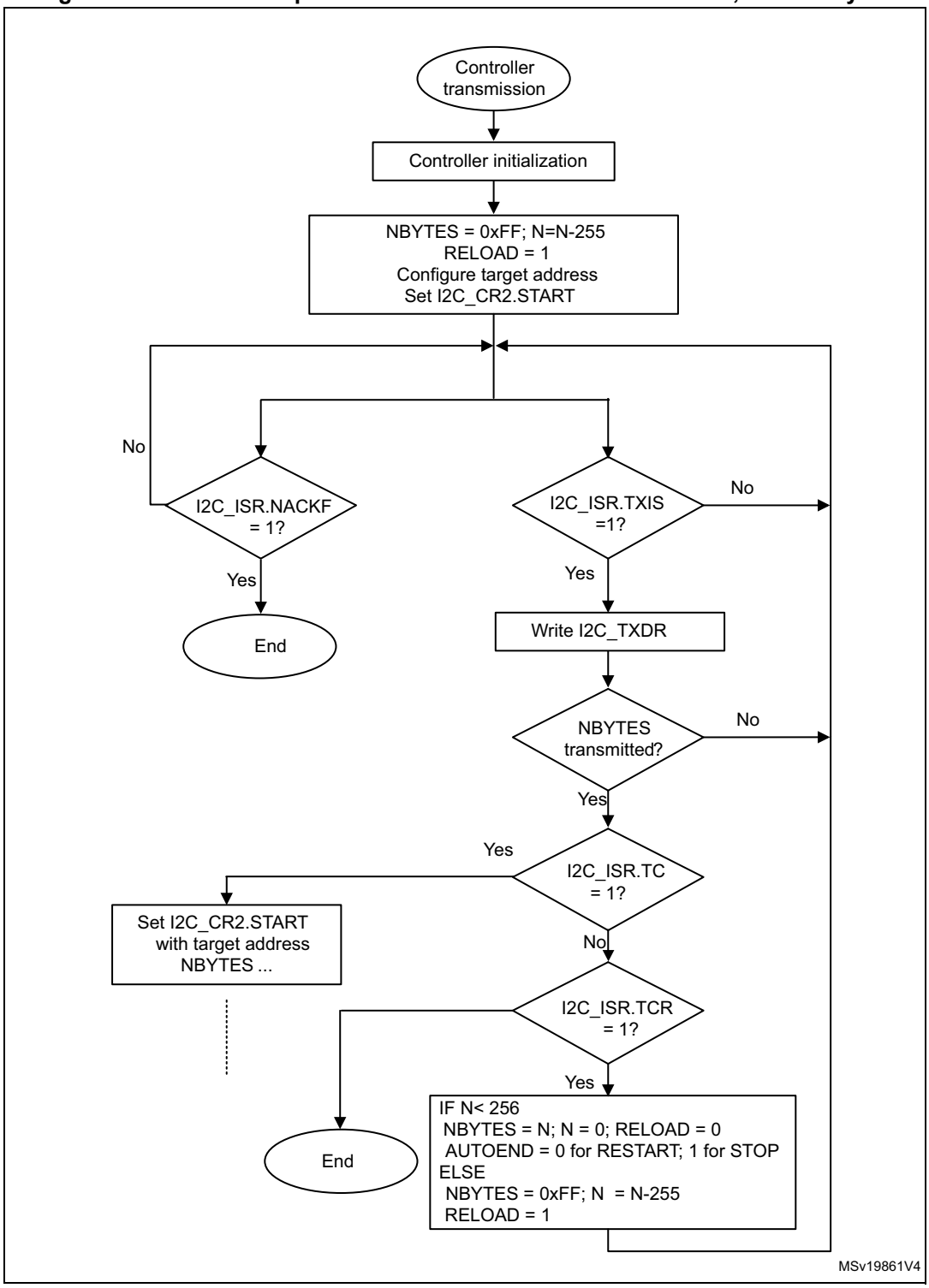
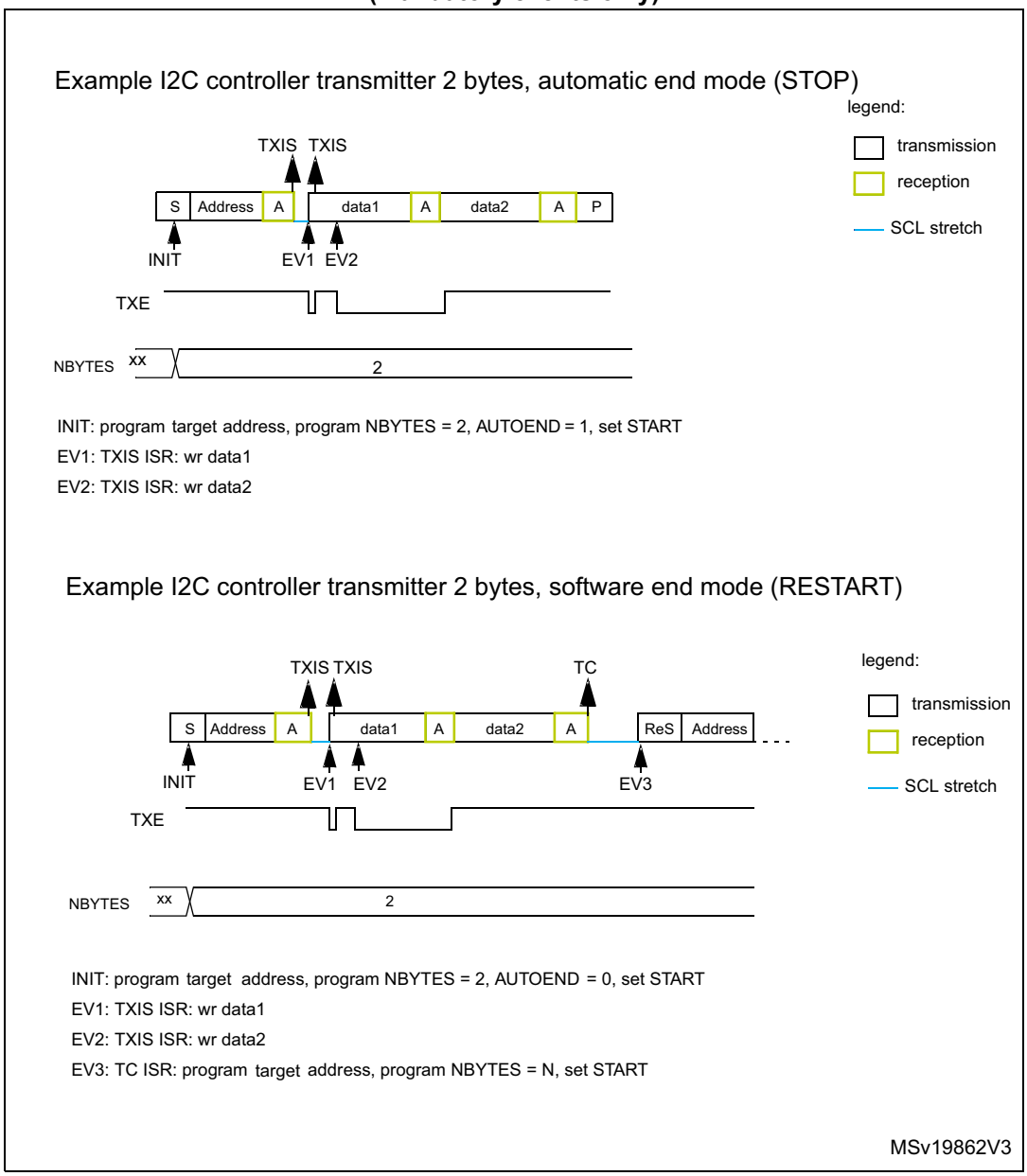


Figure 885. Transfer bus diagrams for I2C controller transmitter (mandatory events only)



Controller receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth SCL pulse. An RXNE event generates an interrupt if the RXIE bit of the I2C_CR1 register is set. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to receive is greater than 255, select the reload mode, by setting the RELOAD bit of the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and he number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a NACK and a STOP are automatically sent after the last received byte.
- In software end mode (AUTOEND = 0), a NACK is automatically sent after the last received byte. The TC flag is set and the SCL line is stretched low in order to allow software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register, with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition and the target address on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

Figure 886. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes

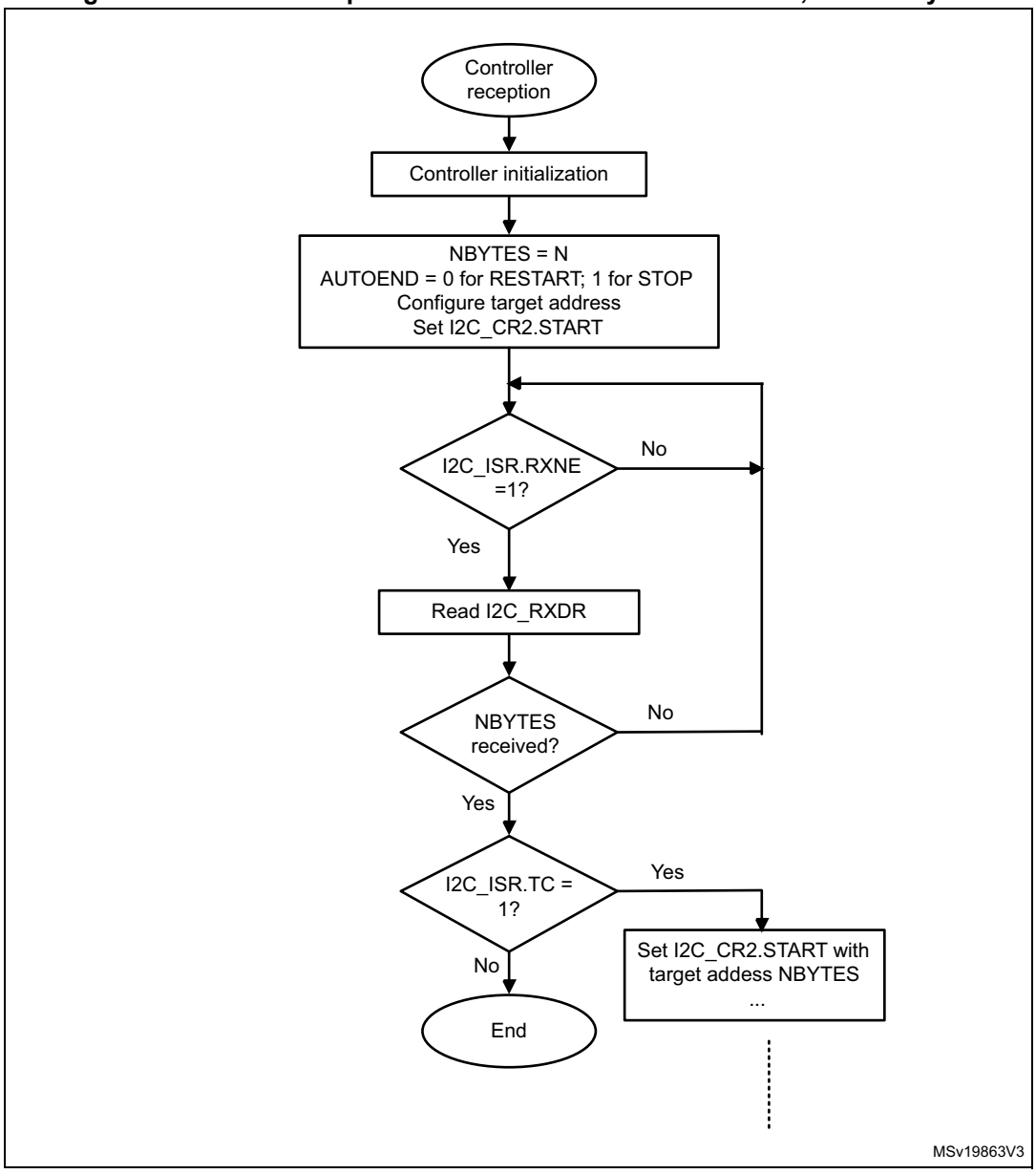


Figure 887. Transfer sequence flow for I2C controller receiver, N > 255 bytes

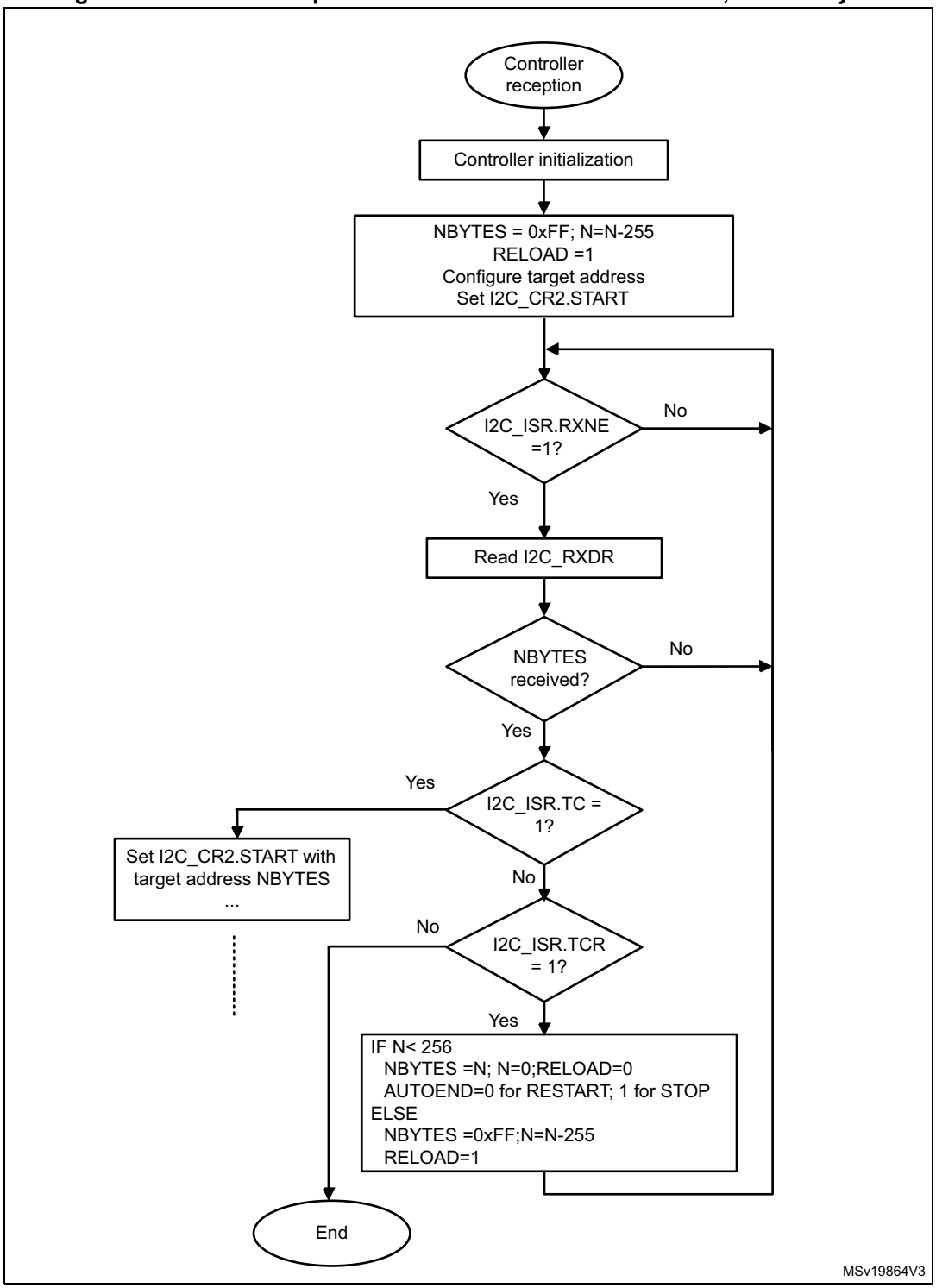
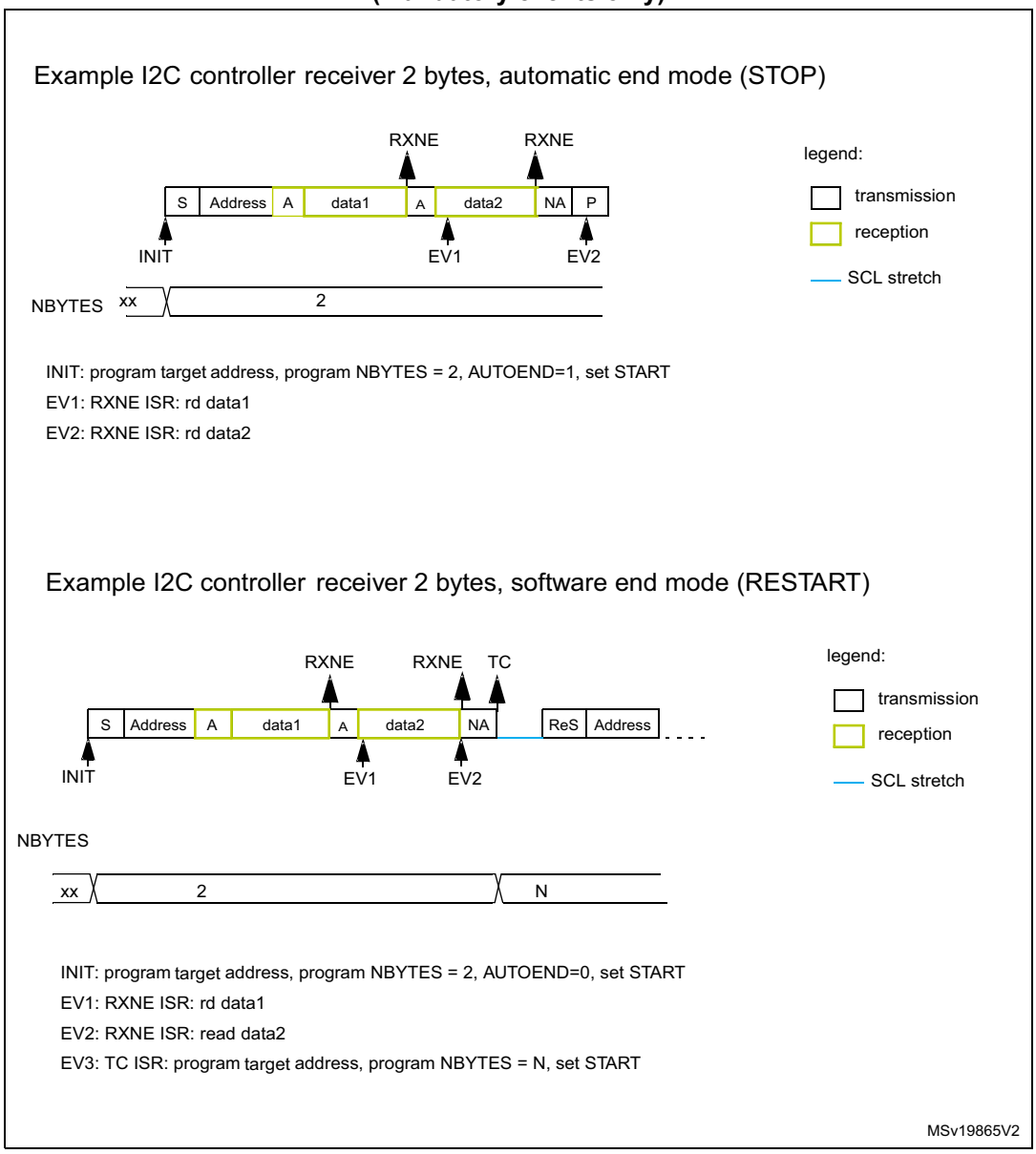


Figure 888. Transfer bus diagrams for I2C controller receiver (mandatory events only)



44.4.10 I2C_TIMINGR register configuration examples

The following tables provide examples of how to program the I2C_TIMINGR register to obtain timings compliant with the I²C-bus specification. To get more accurate configuration values, use the STM32CubeMX tool (*I2C Configuration* window).

Table 506. Timing settings for f_{I2CCLK} of 8 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC[3:0]	0x1	0x1	0x0	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x6
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	7 x 125 ns = 875 ns
SCLH[7:0]	0xC3	0xF	0x3	0x3
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	4 x 125 ns = 500 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~2.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x1	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	1 x 125 ns = 125 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 655 ns.

Table 507. Timing settings for f_{I2CCLK} of 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC[3:0]	0x3	0x3	0x1	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH[7:0]	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~1.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 500 ns.

Table 508. Timing settings for f_{I2CCLK} of 48 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC[3:0]	0xB	0xB	0x5	0x5
SCLL[7:0]	0xC7	0x13	0x9	0x3
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	4 x 125 ns = 500 ns
SCLH[7:0]	0xC3	0xF	0x3	0x1
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~875 ns(4)
SDADEL[3:0]	0x2	0x2	0x3	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	3 x 125 ns = 375 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to the SCL internal detection delay. Values provided for t_{SCL} are only examples.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns
3. t_{SYNC1} + t_{SYNC2} minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns
4. t_{SYNC1} + t_{SYNC2} minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with t_{SYNC1} + t_{SYNC2} = 250 ns

44.4.11 SMBus specific features

Introduction

The system management bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on operation principles of the I²C-bus. The SMBus provides a control bus for system and power management related tasks.

The I2C peripheral is compatible with the SMBus specification (http://smbus.org).

The system management bus specification refers to three types of devices:

- **Target** is a device that receives or responds to a command.
- **Controller** is a device that issues commands, generates clocks, and terminates the transfer.
- **Host** is a specialized controller that provides the main interface to the system CPU. A host must be a controller-target and must support the SMBus *host notify* protocol. Only one host is allowed in a system.

The I2C peripheral can be configured as a controller or a target device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. The device can use any or all of them to communicate. These are: *Quick Command*, *Send Byte*, *Receive Byte*, *Write Byte*, *Write Word*, *Read Byte*, *Read Word*, *Process Call*, *Block Read*, *Block Write*, and *Block Write-Block Read Process Call*. The protocols must be implemented by the user software.

For more details on these protocols, refer to the SMBus specification (http://smbus.org).

STM32CubeMX implements an SMBus stack thanks to X-CUBE-SMBUS, a downloadable software pack that allows basic SMBus configuration per I2C instance.

Address resolution protocol (ARP)

SMBus target address conflicts can be resolved by dynamically assigning a new unique address to each target device. To provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique 128-bit device identifier (UDID). In the I2C peripheral, it is implemented by software.

The I2C peripheral supports the Address resolution protocol (ARP). The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register. The ARP commands must be implemented by the user software.

Arbitration is also performed in target mode for ARP support.

For more details on the SMBus address resolution protocol, refer to the SMBus specification (http://smbus.org).

Received command and data acknowledge control

An SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Host notify protocol

To enable the host notify protocol, set the SMBHEN bit of the I2C_CR1 register. The I2C peripheral then acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a controller and the host as a target.

SMBus alert

The I2C peripheral supports the SMBALERT# optional signal through the SMBA pin. With the SMBALERT# signal, an SMBus target device can signal to the SMBus host that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (0b0001 100). Only the device/devices which pulled SMBALERT# low acknowledges/acknowledge the alert response address.

When the I2C peripheral is configured as an SMBus target device (SMBHEN = 0), the SMBA pin is pulled low by setting the ALERTEN bit of the I2C_CR1 register. The alert response address is enabled at the same time.

When the I2C peripheral is configured as an SMBus host (SMBHEN = 1), the ALERT flag of the I2C_ISR register is set when a falling edge is detected on the SMBA pin and ALERTEN = 1. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. When ALERTEN = 0, the alert line is considered high even if the external SMBA pin is low.

Note: If the SMBus alert pin is not required, keep the ALERTEN bit cleared. The SMBA pin can then be used as a standard GPIO.

Packet error checking

A packet error checking mechanism introduced in the SMBus specification improves reliability and communication robustness. The packet error checking is implemented by

appending a packet error code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x^8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The I2C peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match the hardware calculated PEC.

Timeouts

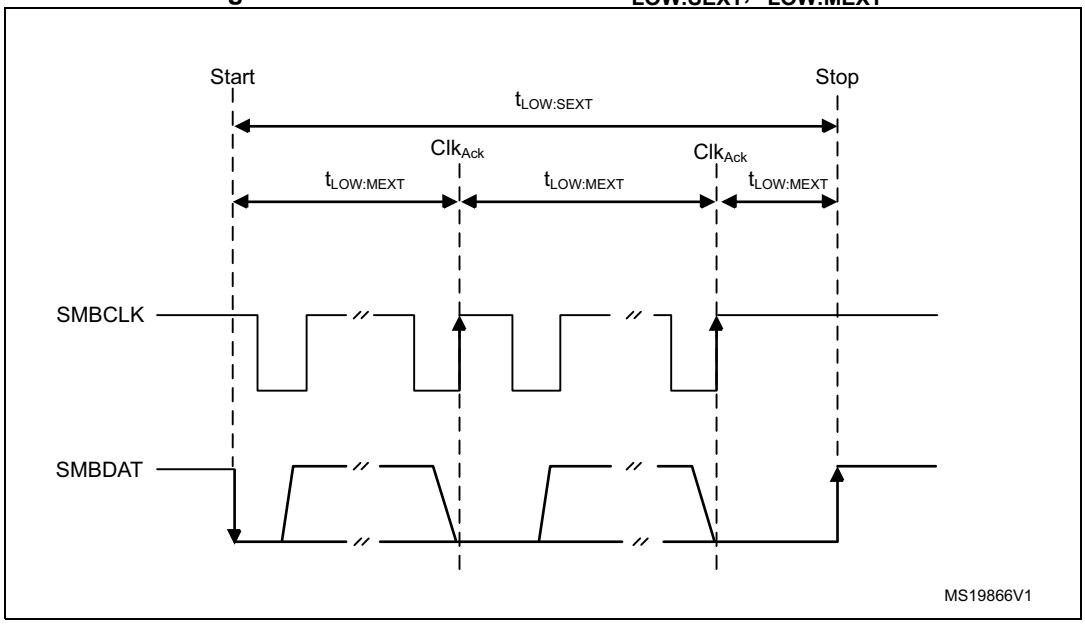
To comply with the SMBus timeout specifications, the I2C peripheral embeds hardware timers.

Table 509. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}$(1)	Cumulative clock low extend time (target device)	-	25	
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (controller device)	-	10	

1. $t_{LOW:SEXT}$ is the cumulative time a given target device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that another target device or the controller also extends the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. The value provided applies to a single target device connected to a full-target controller.
2. $t_{LOW:MEXT}$ is the cumulative time a controller device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a target device or another controller also extends the clock, causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. The value provided applies to a single target device connected to a full-target controller.

Figure 889. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$



Bus idle detection

A controller can assume that the bus is free if it detects that the clock and data signals have been high for $t_{IDLE} > t_{HIGH}$(max) (refer to the table in *Section 44.4.9*).

This timing parameter covers the condition where a controller is dynamically added to the bus, and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the controller must wait long enough to ensure that a transfer is not currently in progress. The I2C peripheral supports a hardware bus idle detection.

44.4.12 SMBus initialization

In addition to the I2C initialization for the I²C-bus, the use of the peripheral for the SMBus communication requires some extra initialization steps.

When initializing I2C for using the device on an SMBus 3.0 compliant bus, it is recommended to enable the corresponding SMB3 bit of the I2C_CR1 register. This shifts the V_{IH} threshold of the corresponding GPIO (configured as GPIO AF) to a lower level.

Received command and data acknowledge control (target mode)

An SMBus receiver must be able to NACK each received command or data. To allow ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Specific addresses (target mode)

The specific SMBus addresses must be enabled if required. Refer to *Bus idle detection* for more details.

The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register.

The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit of the I2C_CR1 register.

The alert response address (0b0001100) is enabled by setting the ALERTEN bit of the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit of the I2C_CR1 register. Then the PEC transfer is managed with the help of the hardware byte counter associated with the NBYTES[7:0] bitfield of the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in target mode. The PEC is transferred after transferring NBYTES[7:0] - 1 data bytes, if the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C peripheral is enabled.

Table 510. SMBus with PEC configuration

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Controller Tx/Rx NBYTES + PEC+ STOP	X	0	1	1
Controller Tx/Rx NBYTES + PEC + ReSTART	X	0	0	1
Target Tx/Rx with PEC	1	0	X	1

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits of the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

$t_{TIMEOUT}$ check

To check the $t_{TIMEOUT}$ parameter, load the 12-bit TIMEOUTA[11:0] bitfield with the timer reload value. Keep the TIDLE bit at 0 to detect the SCL low level timeout.

Then set the TIMOUTEN bit of the I2C_TIMEOUTR register, to enable the timer.

If SCL is tied low for longer than the (TIMEOUTA + 1) x 2048 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 511*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

$t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

A 12-bit timer associated with the TIMEOUTB[11:0] bitfield allows checking $t_{LOW:SEXT}$ for the I2C peripheral operating as a target, or $t_{LOW:MEXT}$ when it operates as a controller. As the standard only specifies a maximum, the user can choose the same value for both. The timer is then enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for longer than the (TIMEOUTB + 1) x 2048 x t_{I2CCLK} period, and within the timeout interval described in *Bus idle detection* section, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 512*.

Caution: Changing the TIMEOUTB[11:0] bitfield value is not allowed when the TEXTEN bit is set.

Bus idle detection

To check the t_{IDLE} period, the TIMEOUTA[11:0] bitfield associated with 12-bit timer must be loaded with the timer reload value. Keep the TIDLE bit at 1 to detect both SCL and SDA high level timeout. Then set the TIMOUTEN bit of the I2C_TIMEOUTR register to enable the timer.

If both the SCL and SDA lines remain high for longer than the (TIMEOUTA + 1) x 4 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 513*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

44.4.13 SMBus I2C_TIMEOUTR register configuration examples

The following tables provide examples of settings to reach desired $t_{TIMEOUT}$, $t_{LOW:SEXT}$, $t_{LOW:MEXT}$, and t_{IDLE} timings at different f_{I2CCLK} frequencies.

Table 511. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms
32 MHz	0x186	0	1	391 x 2048 x 31.25 ns = 25 ms
48 MHz	0x249	0	1	586 x 2048 x 20.08 ns = 25 ms

Table 512. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms

f_{I2CCLK}	TIMEOUTB[11:0]	TEXTEN	$t_{LOW:SEXT}$ $t_{LOW:MEXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms
32 MHz	0x7C	1	125 x 2048 x 31.25 ns = 8 ms
48 MHz	0xBB	1	188 x 2048 x 20.08 ns = 8 ms

Table 513. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	t_{IDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs
32 MHz	0x18F	1	1	400 x 4 x 31.25 ns = 50 µs
48 MHz	0x257	1	1	600 x 4 x 20.08 ns = 50 µs

44.4.14 SMBus target mode

In addition to I2C target transfer management (refer to *Section 44.4.8: I2C target mode*), this section provides extra software flowcharts to support SMBus.

SMBus target transmitter

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case, the total number of TXIS interrupts is NBYTES[7:0] - 1, and the content of the I2C_PECR register is automatically transmitted if the controller requests an extra byte after the transfer of the NBYTES[7:0] - 1 data bytes.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 890. Transfer sequence flow for SMBus target transmitter N bytes + PEC

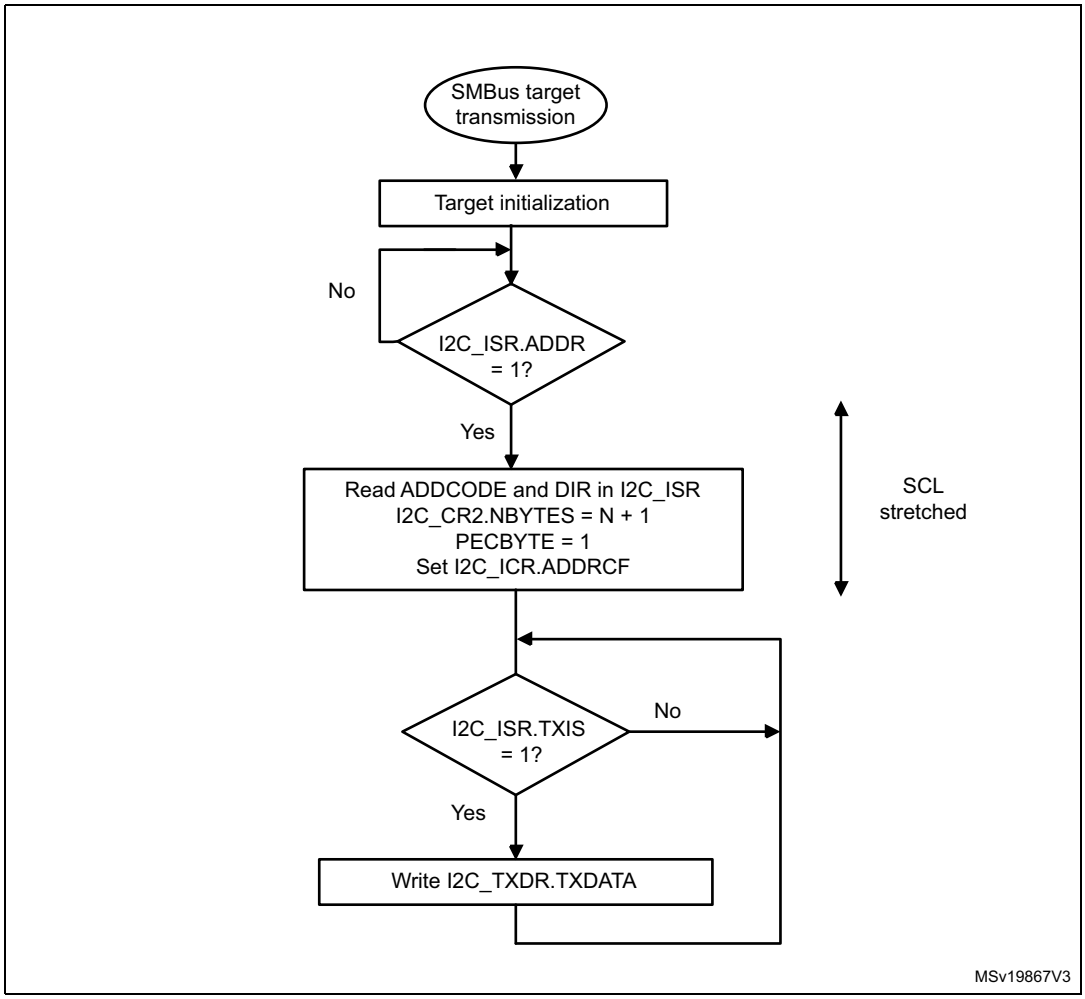
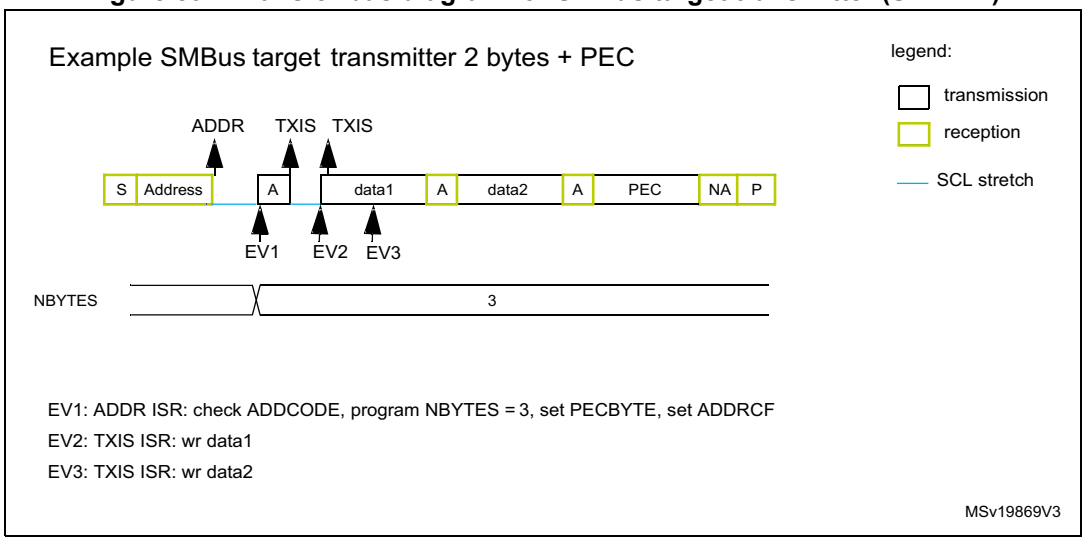


Figure 891. Transfer bus diagram for SMBus target transmitter (SBC = 1)



SMBus target receiver

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC checking at the end of the programmed number of data bytes. To allow the ACK control of each byte, the reload mode must be selected (RELOAD = 1). Refer to *Target byte control mode* for more details.

To check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

Upon a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

If no ACK software control is required, the user can set the PECBYTE bit and, in the same write operation, load NBYTES[7:0] with the number of bytes to receive in a continuous flow. After the receipt of NBYTES[7:0] - 1 bytes, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 892. Transfer sequence flow for SMBus target receiver N bytes + PEC

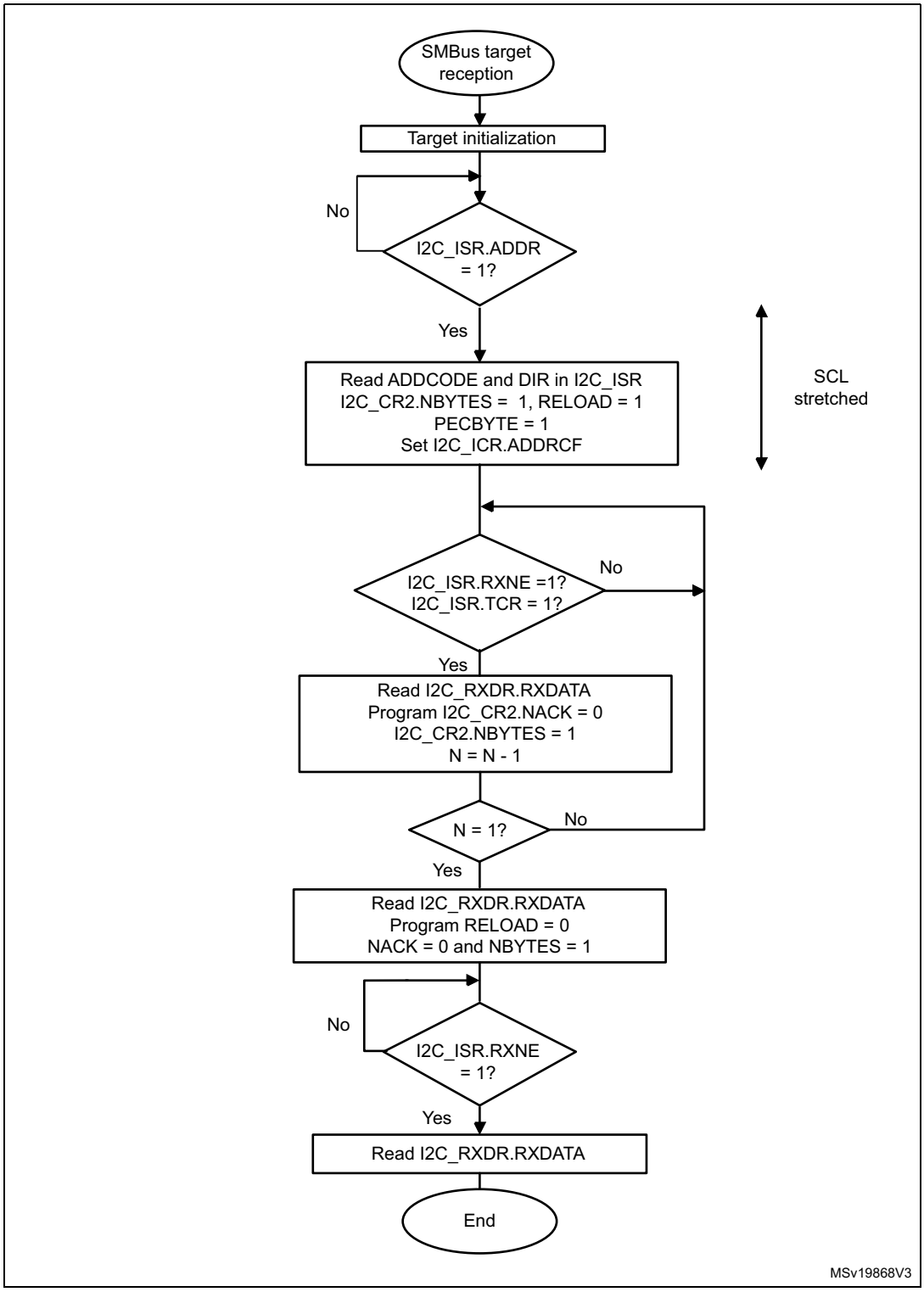
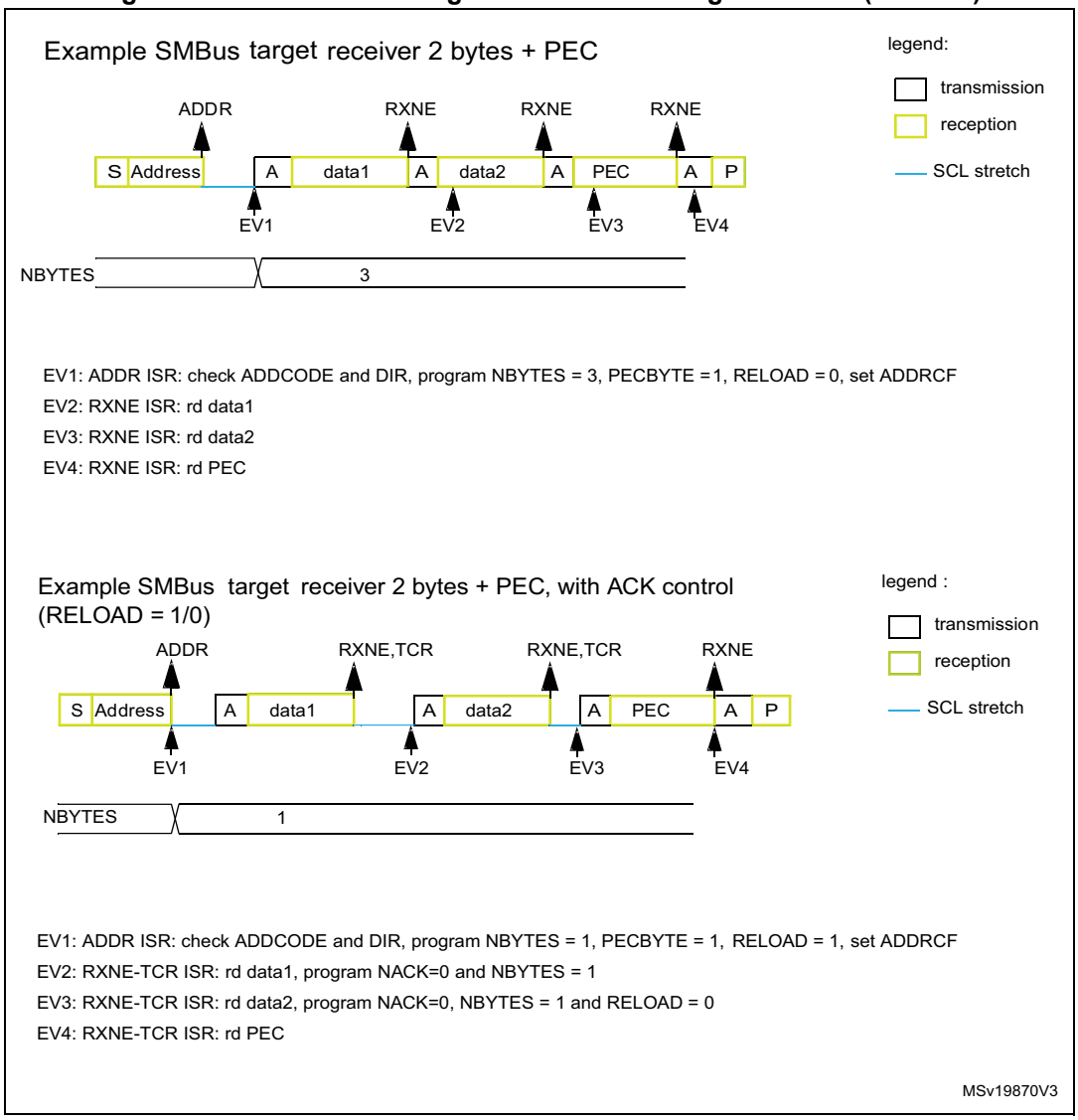


Figure 893. Bus transfer diagrams for SMBus target receiver (SBC = 1)



44.4.15 SMBus controller mode

In addition to I2C controller transfer management (refer to *Section 44.4.9: I2C controller mode*), this section provides extra software flowcharts to support SMBus.

SMBus controller transmitter

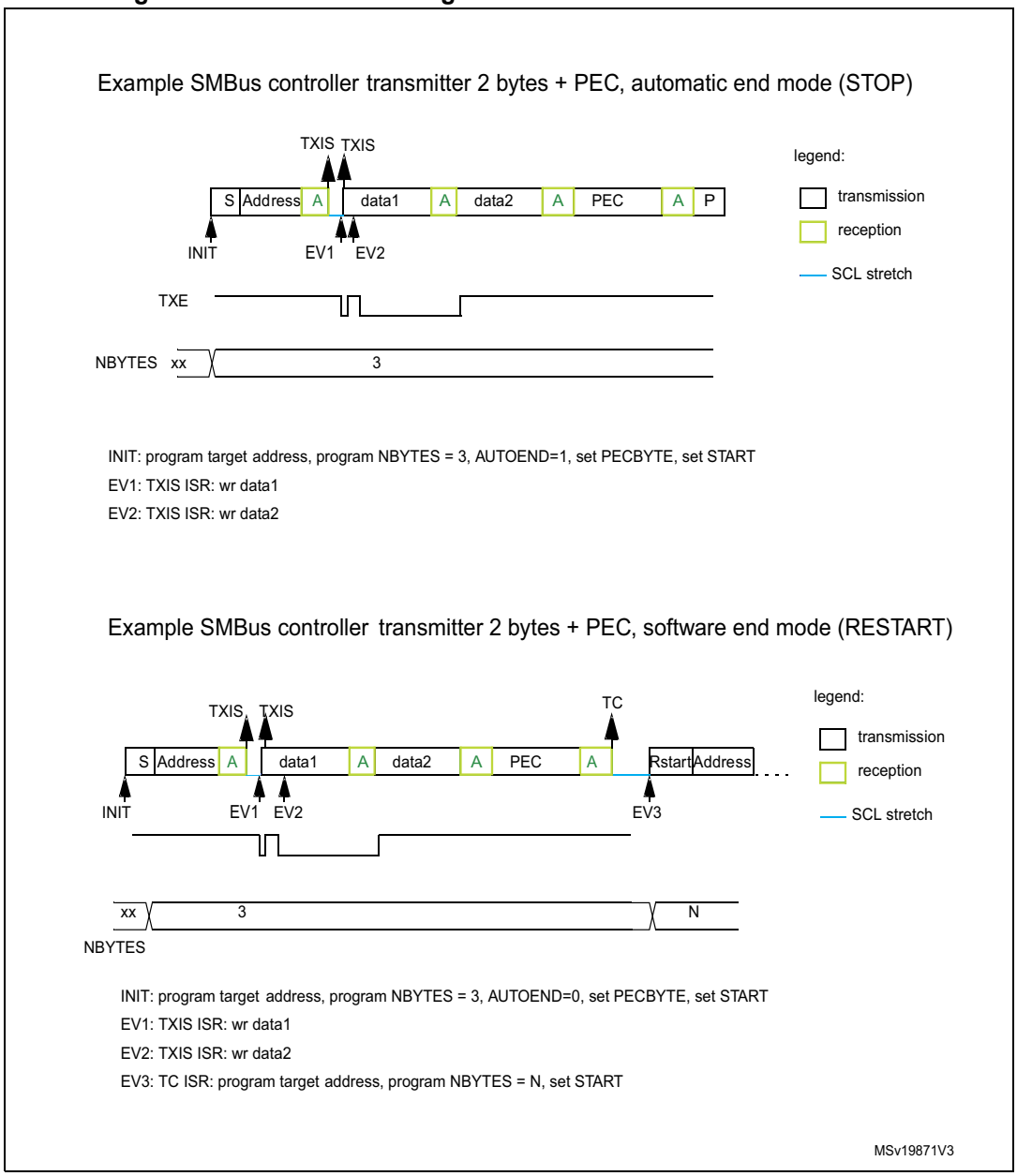
When the SMBus controller wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be loaded in the NBYTES[7:0] bitfield, before setting the START bit. In this case, the total number of TXIS interrupts is NBYTES[7:0] - 1. So if the PECBYTE bit is set when NBYTES[7:0] = 0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus controller wants to send a STOP condition after the PEC, the automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus controller wants to send a RESTART condition after the PEC, the software mode must be selected (AUTOEND = 0). In this case, once NBYTES[7:0] - 1 are transmitted, the I2C_PECR register content is transmitted. The TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 894. Bus transfer diagrams for SMBus controller transmitter



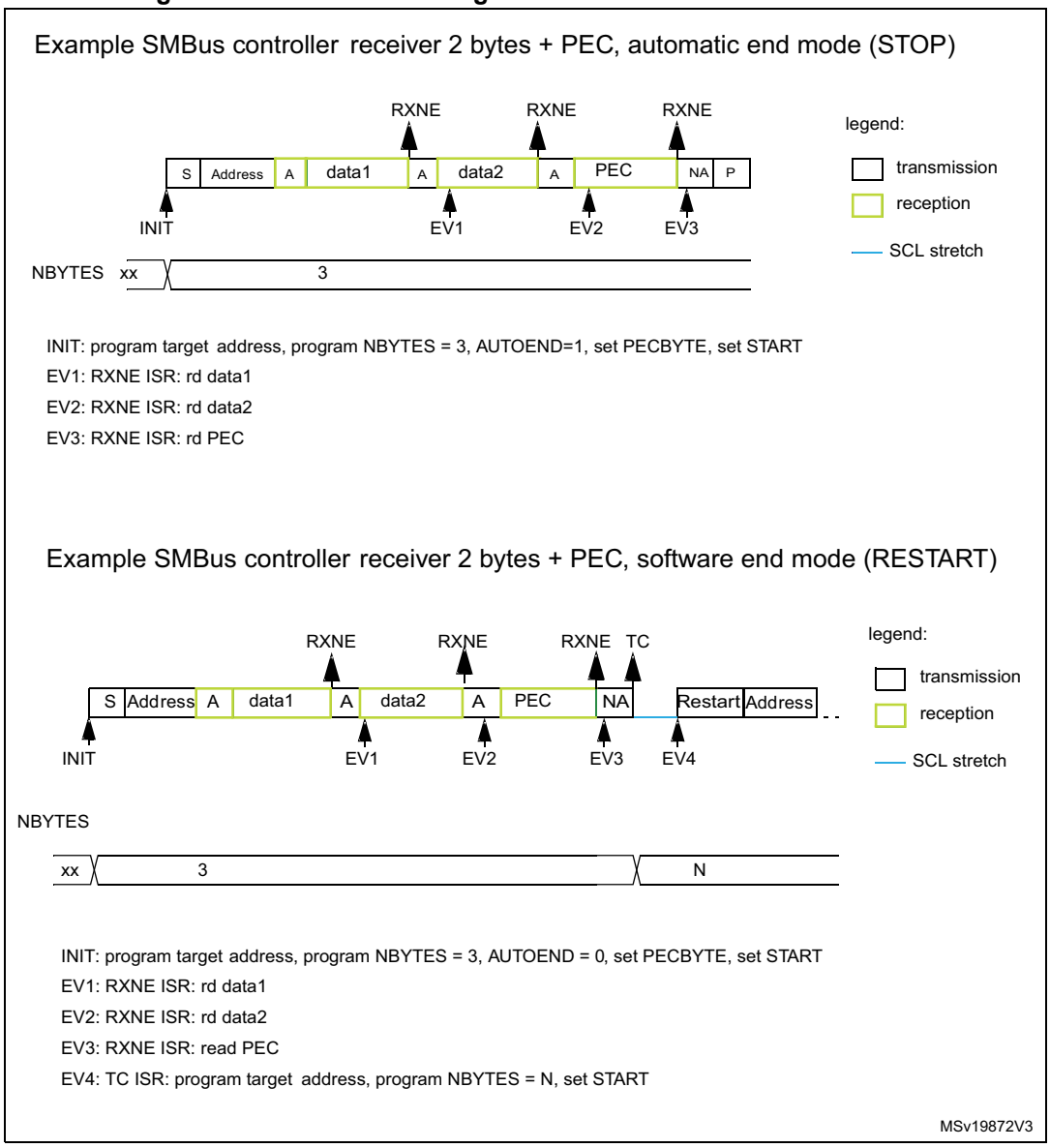
SMBus controller receiver

When the SMBus controller wants to receive, at the end of the transfer, the PEC followed by a STOP condition, the automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus controller receiver wants to receive, at the end of the transfer, the PEC byte followed by a RESTART condition, the software mode must be selected (AUTOEND = 0). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 895. Bus transfer diagrams for SMBus controller receiver



44.4.16 Wake-up from Stop mode on address match

The I2C peripheral is able to wake up the device from Stop mode (APB clock is off), when the device is addressed. All addressing modes are supported.

The wake-up from Stop mode is enabled by setting the WUPEN bit of the I2C_CR1 register. The HSI6 and CSI oscillator must be selected as the clock source for i2c_ker_ck to allow the wake-up from Stop mode.

In Stop mode, the HSI6 and CSI oscillator is stopped. Upon detecting START condition, the I2C interface starts the HSI6 and CSI oscillator and stretches SCL low until the oscillator wakes up.

HSI6 and CSI is then used for the address reception.

If the received address matches the device own address, I2C stretches SCL low until the device wakes up. The stretch is released when the ADDR flag is cleared by software. Then the transfer goes on normally.

If the address does not match, the HSI6 and CSI oscillator is stopped again and the device does not wake up.

Note: When the system clock is used as I2C clock, or when WUPEN = 0, the HSI6 and CSI oscillator does not start upon receiving START condition.

Only an ADDR interrupt can wake the device up. Therefore, do not enter Stop mode when I2C is performing a transfer, either as a controller or as an addressed target after the ADDR flag is set. This can be managed by clearing the SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.

Caution: The digital filter is not compatible with the wake-up from Stop mode feature. Before entering Stop mode with the WUPEN bit set, deactivate the digital filter, by writing zero to the DNF[3:0] bitfield.

Caution: The feature is only available when the HSI6 and CSI oscillator is selected as the I2C clock.

Caution: Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.

Caution: If the wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

44.4.17 Error conditions

The following errors are the conditions that can cause the communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of nine SCL clock pulses. START or STOP condition is detected when an SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C peripheral is involved in the transfer as controller or addressed target (that is, not during the address phase in target mode).

In case of a misplaced START or RESTART detection in target mode, the I2C peripheral enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag of the I2C_ISR register is set, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Arbitration loss (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

In controller mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the controller switches automatically to target mode.

In target mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in target mode when NOSTRETCH = 1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF = 1 and the first data byte must be sent. The content of the I2C_TXDR register is sent if TXE = 0, 0xFF if not.
  - When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Packet error checking error (PECERR)

A PEC error is detected when the received PEC byte does not match the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Timeout error (TIMEOUT)

A timeout error occurs for any of these conditions:

- TIDLE = 0 and SCL remains low for the time defined in the TIMEOUTA[11:0] bitfield: this is used to detect an SMBus timeout.
- TIDLE = 1 and both SDA and SCL remains high for the time defined in the TIMEOUTA [11:0] bitfield: this is used to detect a bus idle condition.
- Controller cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:MEXT}$ parameter).
- Target cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:SEXT}$ parameter).

When a timeout violation is detected in controller mode, a STOP condition is automatically sent.

When a timeout violation is detected in target mode, the SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Alert (ALERT)

The ALERT flag is set when the I2C peripheral is configured as a host (SMBHEN = 1), the SMBALERT# signal detection is enabled (ALERTEN = 1), and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

44.5 I2C in low-power modes

Table 514. Effect of low-power modes to I2C

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Stop(1)	The contents of I2C registers are kept. – WUPEN = 1 and I2C is clocked by an internal oscillator (HSI6 and CSI). The address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. – WUPEN = 0: I2C must be disabled before entering Stop mode.
Standby	The I2C peripheral is powered down. It must be reinitialized after exiting Standby mode.

1. Refer to *Section 44.3: I2C implementation* for information about the Stop modes supported by each instance. If the wake-up from a specific stop mode is not supported, the instance must be disabled before entering that specific Stop mode.

44.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

Table 515. I2C interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Sleep, Stop modes	Exit Standby, Off modes
I2C_EV	Receive buffer not empty	RXNE	RXIE	Read I2C_RXDR register	Yes	No	No
	Transmit buffer interrupt status	TXIS	TXIE	Write I2C_TXDR register			
	STOP detection interrupt flag	STOPF	STOPIE	Write STOPCF = 1			
	Transfer complete reload	TCR	TCIE	Write I2C_CR2 with NBYTES[7:0] ≠ 0			
	Transfer complete	TC		Write START = 1 or STOP = 1			
	Address matched	ADDR	ADDRIE	Write ADDRCF = 1		Yes(1)	
	NACK reception	NACKF	NACKIE	Write NACKCF = 1		No	
I2C_ERR	Bus error	BERR	ERRIE	Write BERRCF = 1	Yes	No	No
	Arbitration loss	ARLO		Write ARLOCF = 1			
	Overrun/underrun	OVR		Write OVRCF = 1			
I2C_ERR	PEC error	PECERR	ERRIE	Write PECERRCF = 1	Yes	No	No
	Timeout/ t_{LOW} error	TIMEOUT		Write TIMEOUTCF = 1			
	SMBus alert	ALERT		Write ALERTCF = 1			

1. The ADDR match event can wake up the device from Stop mode only if the I2C instance supports the wake-up from Stop mode feature. Refer to *Section 44.3: I2C implementation*.

44.7 I2C DMA requests

44.7.1 Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit of the I2C_CR1 register. Data is loaded from an SRAM area configured through the DMA peripheral (see *Section 13: General purpose direct memory access controller (GPDMA)*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software (the transmitted target address cannot be transferred with DMA). When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to *Controller transmitter*.

In target mode:

- With NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
- With NOSTRETCH = 1, the DMA must be initialized before the address match event.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target transmitter* and *SMBus controller transmitter*.

Note: If DMA is used for transmission, it is not required to set the TXIE bit.

44.7.2 Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit of the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured through the DMA peripheral (refer to *Section 13: General purpose direct memory access controller (GPDMA)*) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.

In target mode with NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target receiver* and *SMBus controller receiver*.

Note: If DMA is used for reception, it is not required to set the RXIE bit.

44.8 I2C debug modes

When the device enters debug mode (core halted), the SMBus timeout either continues working normally or stops, depending on the I2Cx bits in the DBG block.

44.9 I2C registers

Refer to *Section 1.2* for the list of abbreviations used in register descriptions.

The registers are accessed by words (32-bit).

44.9.1 I2C control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	SMB3	FMP	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **STOPFACLR**: STOP detection flag (STOPF) automatic clear

0: STOPF flag is set by hardware, cleared by software by setting STOPCF bit.

1: STOPF flag remains cleared by hardware. This mode can be used in NOSTRETCH target mode, to avoid the overrun error if the STOPF flag is not cleared before next data transmission. This allows a target data management by DMA only, without any interrupt from peripheral.

Bit 30 **ADDRACLR**: Address match flag (ADDR) automatic clear

0: ADDR flag is set by hardware, cleared by software by setting ADDRCF bit.

1: ADDR flag remains cleared by hardware. This mode can be used in target mode, to avoid the ADDR clock stretching if the I2C enables only one target address. This allows a target data management by DMA only, without any interrupt from peripheral.

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **SMB3**: SMBus specification 3.0 I/O compliance enable

This bit only affects I/Os with "_b" structure. Refer to the product datasheet, pin definition table. Setting the bit enables the SCL, SDA, and SMBA I/Os of "_b" structure to comply with the version 3.0 of SMBus specification, with regard to the V_{IL}, V_{IH} and V_{HYS} parameters. When the bit is cleared, the I/Os of "_b" structure behave as standard GPIOs.

0: SMBus specification 3.0 I/Os compliance disabled

1: SMBus specification 3.0 I/Os compliance enabled

Bit 24 **FMP**: Fast-mode Plus drive enable
0: FMP I/O drive disabled
1: FMP I/O drive enabled

Bit 23 **PECEN:** PEC enable
0: PEC calculation disabled
1: PEC calculation enabled

Bit 22 **ALERTEN**: SMBus alert enable
0: The SMBALERT# signal on SMBA pin is not supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is released and the alert response address header is disabled (0001100x followed by NACK).
1: The SMBALERT# signal on SMBA pin is supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is driven low and the alert response address header is enabled (0001100x followed by ACK).
Note: When ALERTEN = 0, the SMBA pin can be used as a standard GPIO.

Bit 21 **SMBDEN**: SMBus device default address enable
0: Device default address disabled. Address 0b1100001x is NACKed.
1: Device default address enabled. Address 0b1100001x is ACKed.

Bit 20 **SMBHEN**: SMBus host address enable
0: Host address disabled. Address 0b0001000x is NACKed.
1: Host address enabled. Address 0b0001000x is ACKed.

Bit 19 **GCEN**: General call enable
0: General call disabled. Address 0b00000000 is NACKed.
1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 **WUPEN**: Wake-up from Stop mode enable
0: Wake-up from Stop mode disabled.
1: Wake-up from Stop mode enabled.
Note: WUPEN can be set only when DNF[3:0] = 0000.

Bit 17 **NOSTRETCH**: Clock stretching disable
This bit is used to disable clock stretching in target mode. It must be kept cleared in controller mode.
0: Clock stretching enabled
1: Clock stretching disabled
Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 16 **SBC**: Target byte control
This bit is used to enable hardware byte control in target mode.
0: Target byte control disabled
1: Target byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable
0: DMA mode disabled for reception
1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable
0: DMA mode disabled for transmission
1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF:** Analog noise filter OFF
0: Analog noise filter enabled
1: Analog noise filter disabled
Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bits 11:8 **DNF[3:0]**: Digital noise filter
These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to DNF[3:0] * t_{I2CCLK}
0000: Digital filter disabled
0001: Digital filter enabled and filtering capability up to one t_{I2CCLK}
...
1111: digital filter enabled and filtering capability up to fifteen t_{I2CCLK}
Note: If the analog filter is enabled, the digital filter is added to it. This filter can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 7 **ERRIE**: Error interrupts enable
0: Error detection interrupts disabled
1: Error detection interrupts enabled
Note: Any of these errors generates an interrupt:
- arbitration loss (ARLO)
- bus error detection (BERR)
- overrun/underrun (OVR)
- timeout detection (TIMEOUT)
- PEC error detection (PECERR)
- alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer complete interrupt enable
0: Transfer complete interrupt disabled
1: Transfer complete interrupt enabled
Note: Any of these events generates an interrupt:
Transfer complete (TC)
Transfer complete reload (TCR)

Bit 5 **STOPIE**: STOP detection interrupt enable
0: STOP detection (STOPF) interrupt disabled
1: STOP detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received interrupt enable
0: Not acknowledge (NACKF) received interrupts disabled
1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match interrupt enable (target only)
0: Address match (ADDR) interrupts disabled
1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX interrupt enable
0: Receive (RXNE) interrupt disabled
1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX interrupt enable
0: Transmit (TXIS) interrupt disabled
1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable
0: Peripheral disabled
1: Peripheral enabled
Note: When PE = 0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least three APB clock cycles.

44.9.2 I2C control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]							
					rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 26 **PECBYTE**: Packet error checking byte
This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE = 0.
0: No PEC transfer
1: PEC transmission/reception is requested
Note: Writing 0 to this bit has no effect.
This bit has no effect when RELOAD is set, and in target mode when SBC = 0.

Bit 25 **AUTOEND**: Automatic end mode (controller mode)
This bit is set and cleared by software.
0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.
1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.
Note: This bit has no effect in target mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode
This bit is set and cleared by software.
0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).
1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes
The number of bytes to be transmitted/received is programmed there. This field is don't care in target mode with SBC = 0.
Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (target mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE = 0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

Note: Writing 0 to this bit has no effect.

This bit is used only in target mode: in controller receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in target receiver NOSTRETCH mode, a NACK is automatically generated, whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE = 1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: STOP condition generation

This bit only pertains to controller mode. It is set by software and cleared by hardware when a STOP condition is detected or when PE = 0.

0: No STOP generation

1: STOP generation after current byte transfer

Note: Writing 0 to this bit has no effect.

Bit 13 **START**: START condition generation

This bit is set by software. It is cleared by hardware after the START condition followed by the address sequence is sent, by an arbitration loss, by an address matched in target mode, by a timeout error detection, or when PE = 0.

0: No START generation

1: RESTART/START generation:

If the I2C is already in controller mode with AUTOEND = 0, setting this bit generates a repeated START condition when RELOAD = 0, after the end of the NBYTES transfer.

Otherwise, setting this bit generates a START condition once the bus is free.

Note: Writing 0 to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in target mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (controller receiver mode)

0: The controller sends the complete 10-bit target address read sequence: START + 2 bytes 10-bit address in write direction + RESTART + first seven bits of the 10-bit address in read direction.

1: The controller sends only the first seven bits of the 10-bit address, followed by read direction.

Note: Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (controller mode)

0: The controller operates in 7-bit addressing mode

1: The controller operates in 10-bit addressing mode

Note: Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (controller mode)

0: Controller requests a write transfer

1: Controller requests a read transfer

Note: Changing this bit when the START bit is set is not allowed.

Bits 9:0 **SADD[9:0]**: Target address (controller mode)

Condition: In 7-bit addressing mode (ADD10 = 0):
SADD[7:1] must be written with the 7-bit target address to be sent. Bits SADD[9], SADD[8] and SADD[0] are don't care.

Condition: In 10-bit addressing mode (ADD10 = 1):
SADD[9:0] must be written with the 10-bit target address to be sent.

Note: Changing these bits when the START bit is set is not allowed.

44.9.3 I2C own address 1 register (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA1EN**: Own address 1 enable

0: Own address 1 disabled. The received target address OA1 is NACKed.
1: Own address 1 enabled. The received target address OA1 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **OA1MODE**: Own address 1 10-bit mode

0: Own address 1 is a 7-bit address.
1: Own address 1 is a 10-bit address.

Note: This bit can be written only when OA1EN = 0.

Bits 9:0 **OA1[9:0]**: Interface own target address

7-bit addressing mode: OA1[7:1] contains the 7-bit own target address. Bits OA1[9], OA1[8] and OA1[0] are don't care.
10-bit addressing mode: OA1[9:0] contains the 10-bit own target address.

Note: These bits can be written only when OA1EN = 0.

44.9.4 I2C own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA2EN**: Own address 2 enable
0: Own address 2 disabled. The received target address OA2 is NACKed.
1: Own address 2 enabled. The received target address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own address 2 masks
000: No mask
001: OA2[1] is masked and don't care. Only OA2[7:2] are compared.
010: OA2[2:1] are masked and don't care. Only OA2[7:3] are compared.
011: OA2[3:1] are masked and don't care. Only OA2[7:4] are compared.
100: OA2[4:1] are masked and don't care. Only OA2[7:5] are compared.
101: OA2[5:1] are masked and don't care. Only OA2[7:6] are compared.
110: OA2[6:1] are masked and don't care. Only OA2[7] is compared.
111: OA2[7:1] are masked and don't care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

Note: These bits can be written only when OA2EN = 0.
As soon as OA2MSK ≠ 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged, even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address
7-bit addressing mode: 7-bit address

Note: These bits can be written only when OA2EN = 0.

Bit 0 Reserved, must be kept at reset value.

44.9.5 I2C timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: no wait states

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

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale i2c_ker_ck to generate the clock period t_{PRESC} used for data setup and hold counters (refer to section *I2C timings*), and for SCL high and low level counters (refer to section *I2C controller initialization*).

t_{PRESC} = (PRESC + 1) x t_{I2CCLK}

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} = (SCLDEL + 1) x t_{PRESC} between SDA edge and SCL rising edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SCLDEL}.

Note: t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge and SDA edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SDADEL}.

t_{SDADEL}= SDADEL x t_{PRESC}

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (controller mode)

This field is used to generate the SCL high period in controller mode.

t_{SCLH} = (SCLH + 1) x t_{PRESC}

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (controller mode)

This field is used to generate the SCL low period in controller mode.

t_{SCLL} = (SCLL + 1) x t_{PRESC}

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C peripheral is disabled (PE = 0).

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

44.9.6 I2C timeout register (I2C_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

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

Bit 31 **TEXTEN**: Extended clock timeout enable

0: Extended clock timeout detection is disabled

1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{LOW:EXT}$ is done by the I2C interface, a timeout error is detected (TIMEOUT = 1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B

This field is used to configure the cumulative clock extension timeout:

– Controller mode: the controller cumulative clock low extend time ($t_{LOW:MEXT}$) is detected

– Target mode: the target cumulative clock low extend time ($t_{LOW:SEXT}$) is detected

$t_{LOW:EXT}$ = (TIMEOUTB + TIDLE = 01) x 2048 x t_{I2CCLK}

Note: These bits can be written only when TEXTEN = 0.

Bit 15 **TIMOUTEN**: Clock timeout enable

0: SCL timeout detection is disabled

1: SCL timeout detection is enabled. When SCL is low for more than $t_{TIMEOUT}$ (TIDLE = 0) or high for more than t_{IDLE} (TIDLE = 1), a timeout error is detected (TIMEOUT = 1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection

0: TIMEOUTA is used to detect SCL low timeout

1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)

Note: This bit can be written only when TIMOUTEN = 0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus timeout A

This field is used to configure:

The SCL low timeout condition $t_{TIMEOUT}$ when TIDLE = 0

$t_{TIMEOUT}$= (TIMEOUTA + 1) x 2048 x t_{I2CCLK}

The bus idle condition (both SCL and SDA high) when TIDLE = 1

t_{IDLE}= (TIMEOUTA + 1) x 4 x t_{I2CCLK}

Note: These bits can be written only when TIMOUTEN = 0.

44.9.7 I2C interrupt and status register (I2C_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **ADDCODE[6:0]**: Address match code (target mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1). In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the two MSBs of the address.

Bit 16 **DIR**: Transfer direction (target mode)

This flag is updated when an address match event occurs (ADDR = 1).

0: Write transfer, target enters receiver mode.

1: Read transfer, target enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected, and cleared by hardware when a STOP condition is detected, or when PE = 0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN = 1 (SMBus host configuration), ALERTEN = 1 and an SMBALERT# event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 12 **TIMEOUT**: Timeout or t_{LOW} detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 11 **PECERR**: PEC error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 10 **OVR**: Overrun/underrun (target mode)

This flag is set by hardware in target mode with NOSTRETCH = 1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 8 **BERR**: Bus error

This flag is set by hardware when a misplaced START or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in target mode. It is cleared by software by setting the BERRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 7 **TCR**: Transfer complete reload

This flag is set by hardware when RELOAD = 1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

Note: This bit is cleared by hardware when PE = 0.

This flag is only for controller mode, or for target mode when the SBC bit is set.

Bit 6 **TC**: Transfer complete (controller mode)

This flag is set by hardware when RELOAD = 0, AUTOEND = 0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

Note: This bit is cleared by hardware when PE = 0.

Bit 5 **STOPF**: STOP detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

- as a controller, provided that the STOP condition is generated by the peripheral.
- as a target, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 4 **NACKF**: Not acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 3 **ADDR**: Address matched (target mode)

This bit is set by hardware as soon as the received target address matched with one of the enabled target addresses. It is cleared by software by setting *ADDRCF bit.*

Note: This bit is cleared by hardware when PE = 0.

Bit 2 **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

Note: This bit is cleared by hardware when PE = 0.

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software only when NOSTRETCH = 1, to generate a TXIS event (interrupt if TXIE = 1 or DMA request if TXDMAEN = 1).

Note: This bit is cleared by hardware when PE = 0.

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software in order to flush the transmit data register I2C_TXDR.

Note: This bit is set by hardware when PE = 0.

44.9.8 I2C interrupt clear register (I2C_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **ALERTCF**: Alert flag clear
Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

Bit 12 **TIMOUTCF**: Timeout detection flag clear
Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Bit 11 **PECCF**: PEC error flag clear
Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

Bit 10 **OVRCF**: Overrun/underrun flag clear
Writing 1 to this bit clears the OVR flag in the I2C_ISR register.

Bit 9 **ARLOCF**: Arbitration lost flag clear
Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear
Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STOPCF**: STOP detection flag clear
Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not acknowledge flag clear
Writing 1 to this bit clears the NACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear
Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

44.9.9 I2C PEC register (I2C_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: no wait states

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

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

Bits 7:0 **PEC[7:0]:** Packet error checking register
This field contains the internal PEC when PECEN=1.
The PEC is cleared by hardware when PE = 0.

44.9.10 I2C receive data register (I2C_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: no wait states

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

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

Bits 7:0 **RXDATA[7:0]:** 8-bit receive data
Data byte received from the I²C-bus.

44.9.11 I2C transmit data register (I2C_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: no wait states

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

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

Bits 7:0 **TXDATA[7:0]:** 8-bit transmit data

Data byte to be transmitted to the I²C-bus

Note: These bits can be written only when TXE = 1.

44.9.12 I2C register map

The table below provides the I2C register map and the reset values.

Table 516. I2C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1	STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	SMB3	FMP	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value	0	0					0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	I2C_CR2	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	I2C_OAR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0
0x0C	I2C_OAR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK [2:0]			OA2[7:1]							Res.
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	
0x10	I2C_TIMINGR	PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL [3:0]				SDADEL [3:0]				SCLH[7:0]								SCLL[7:0]							
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	I2C_TIMEOUTR	TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]												TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	I2C_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	I2C_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	I2C_PECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x24	I2C_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	I2C_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

45 Improved inter-integrated circuit (I3C)

45.1 I3C introduction

The I3C interface handles communication with other devices such as sensor(s) and host processor.

An I3C bus is a two-wire, serial single-ended, multi-drop bus, intended to improve the legacy I^2C bus. The I3C interface improves the features of the I^2C interface (adding especially in-band interrupts, dynamic address assignment, energy-efficient data rates, power management) meanwhile preserving some backward compatibility: it allows an I^2C target to operate on an I3C bus in legacy I^2C standard-mode (Sm), fast-mode (Fm), or legacy I^2C fast-mode plus (Fm+), provided that the latter does not perform clock stretching.

The I3C SDR-only peripheral implements all the required features and many optional features of the MIPI® I3C specification v1.1 (see *Table 517: I3C peripheral controller/target features versus MIPI v1.1*). It can control all I3C bus-specific sequencing, protocol, arbitration, and timing.

After reset, the software can initialize the I3C peripheral as primary controller or as target:

- As I3C controller, the software can enable the I3C peripheral and initialize the I3C bus.
- As I3C target, the software can enable the I3C peripheral to operate on an I3C bus or an I^2C bus, provided that the I^2C bus is restricted to the I^2C sub-features that are compliant with a MIPI I3C bus protocol: only Sm, Fm or Fm+ speed, no clock stretching, no multiple 7-bit addresses, no general call message, no 10-bit addressing, no SMBus.

The I3C peripheral can be used with DMA, to off-load the CPU.

45.2 I3C main features

The I3C peripheral supports:

- I^2C-bus specification rev7, as I3C target with 7-bit I^2C static address, optional 50 ns analog spike filter and optional 20 mA open drain driver, provided that the I^2C bus is restricted to the following set of features:
  - Standard-mode (Sm) up to 100 kHz
  - Fast-mode (Fm) up to 400 kHz
  - Fast-mode plus (Fm+) up to 1 MHz
  - no clock stretching, no general call message, no multiple 7-bit addresses, no 10-bit addressing mode, no programmable digital noise filter, no SMBus.
- MIPI® I3C specification v1.1 (see details in *Table 517*), as:
  - I3C SDR-only primary controller
  - I3C SDR-only secondary controller
  - I3C SDR-only target, possibly initialized with a 7-bit I^2C static address
- I3C SCL bus clock frequency up to 12.5 MHz
- Registers configuration from the host application via the APB slave port
- Queued data transfers:

  - Transmit FIFO (TX-FIFO) for data bytes/words to be transmitted on the I3C/I^2C bus
  - Receive FIFO (RX-FIFO) for received data bytes/words on the I3C/I^2C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Queued control/status transfers, when controller:
  - Control FIFO (C-FIFO) for control words to be sent on the I3C bus
  - Optional status FIFO (S-FIFO) for status words as received on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Messages:
  - Legacy I^2C read/write messages in Sm/Fm/Fm+
  - I3C SDR read/write private messages
  - I3C SDR broadcast CCC messages (see details in *Table 527*)
  - I3C SDR read/write direct CCC messages (see details in *Table 527*)
- Frame-level management, when controller:
  - Optional C-FIFO and TX-FIFO preload
  - Multiple messages encapsulation
  - Optional arbitrable header generation on the I3C bus
  - HDR exit pattern generation on the I3C bus for error recovery
- Programmable bus timing, when controller:
  - SCL high and low period
  - SDA hold time
  - Bus free (minimum) time
  - Bus available/idle condition time
  - Clock stall time
- Target-initiated requests management:
  - Simultaneous support up to four I3C targets, when controller
  - In-band interrupts, with programmable IBI payload (up to 4 bytes), with pending read notification support
  - Bus control request, with recovery flow support and hand-off delay
  - Hot-join mechanism
- HDR exit pattern detection, when target
- Bus error management:
  - CEx with x = 0, 1, 2, 3 when controller
  - TEx with x = 0, 1, ... , 6 when target
  - Bus control switch error and recovery
  - Target reset
- Individual programmable event-based management:
  - Per-event identification with flag reporting and clear control
  - Host application notification via flag polling, and/or via interrupt with a per-event programmable enable
  - Error type identification
- Wake-up from Stop mode(s), as controller (see *Section 45.3.2*):

  - On an in-band interrupt without payload
  - On a hot-join request
  - On a controller-role request
- Wake-up from Stop mode(s), as target (see *Section 45.3.2*):
  - On a reset pattern
  - On a missed start on the serial bus
- Multi-clock domain management:
  - Separate APB clock and kernel clock, driven from independently programmed clock sources via the RCC, in addition to SCL clock
  - Minimum operating frequency for the kernel clock and the APB clock vs. the application-driven SCL clock (see clocks constraints in *Section 45.6.2*)

45.2.1 I3C MIPI® support

The I3C peripheral supports the MIPI specification v1.1, as defined in *Table 517*.

Table 517. I3C peripheral controller/target features versus MIPI v1.1

Feature	MIPI I3C v1.1	I3C peripheral		Comments
		When controller	When target	
I3C SDR message	X	X	X	-
Legacy I^2C message (Sm/Fm/Fm+)	X	X	X	Mandatory when controller if the I3C bus is mixed with (external) legacy I^2C target(s). Optional in MIPI v1.1 when target. Supported when I3C target on an I^2C bus or an I3C bus.
HDR DDR message	X	-	-	Optional in MIPI v1.1
HDR-TSL/TSP, HDR-BT	X	-	-	Optional in MIPI v1.1
Dynamic address assignment	X	X	X	Supported broadcast ENTDAA, SETAASA and direct SETDASA when controller and target.
7-bit static I^2C address	X	X	X	Supported when I3C target on an I^2C bus. May be used at initialization phase when I3C target on an I3C bus.
Grouped addressing	X	X	-	Optional in MIPI v1.1
CCCs	X	X	X	Mandatory and some optional CCCs supported (refer to *Table 527* when controller/target).
Error detection and recovery	X	X	X	-
In-band interrupt (with MDB)	X	X	X	-
Secondary controller	X	X	X	-
Hot-join mechanism	X	X	X	-
Target reset	X	X	X	-
Synchronous timing control	X	X	-	Optional in MIPI v1.1
Asynchronous timing control 0	X	X	-	Mandatory in MIPI v1.1 when controller (refer to *Timing control and sensor acquisition timestamping when I3C controller*). Optional in MIPI v1.1 when target.
Asynchronous timing control 1, 2, 3	X	-	-	Optional in MIPI v1.1
Device to device tunneling	X	X	-	Optional in MIPI v1.1
Multi-lane data transfer	X	-	-	Optional in MIPI v1.1
Monitoring device early termination	X	-	-	Optional in MIPI v1.1

45.2.2 I3C without clock stretching by the target

The I3C bus is intended to deprecate the I²C bus. The I²C specification optionally features clock stretching by the target, but this is not allowed by the MIPI® I3C specification.

The user may have to port this I²C feature when the I²C bus is replaced by an I3C bus. This section details four main alternative methods to replace clock stretching by the target for the same applicative purpose.

The I²C SCL is open drain. (Optional) I2C clock stretching by the target pauses a transaction by holding the SCL line low:

- It is generally used at byte level, when the target software needs more time to store a received byte or to prepare another byte for transmission.
- It may be used at bit level, typically by an MCU without an I2C bus for software emulation of I²C SCL and SDA lines over GPIOs.

The main advantage of this feature is on the target side; it is convenient for the application software of the target device to have direct control on the I²C bus, and to suspend an I²C transaction when it needs more time before answering.

This feature has a disadvantage on the bus and controller side since there is no suspension timeout in the I²C specification. The I²C bus and controller can become permanently hanged by a lost target. There is no in-band timeout. Bus error recovery is not possible without another alert signal (for example SMBus).

This I²C target capability is not allowed by the MIPI® I3C specification when a target is operating on theI3C bus: the I3C SCL line is driven in push-pull mode by the controller.

Note: The I3C SDA line is a mix of push-pull and open-drain phases.

MIPI® I3C relies on the following four alternative methods to guarantee that a target always has enough time to respond to an I3C or legacy I²C message:

- *I3C byte-level a read data ACK/NACK*
- *I3C/I2C target needs a clock stalling of up to 100 µs (at different phases)*
- *IBI with pending read: for read, if I3C target needs a clock stalling above 100 µs*
- *Maximum read turnaround time: for read, if I3C target needs a clock stalling above 100 µs*

I3C byte-level a read data ACK/NACK

An I3C target can ACK/NACK at byte level a read data

- Any controller or addressed target (new feature in I3C) can terminate the number of exchanged data at a byte level.
- A target needing more time can terminate the transfer early by driving SDA low on the 9th T bit.
- The controller can again ask for new bytes to be read if more bytes are expected from the target.

This is the contrary of I²C where only the controller can terminate a read transfer.

- The I²C controller cannot distinguish whether or not a data byte was transmitted by a (slow) target.
- To prevent this I²C protocol issue, the I²C target must be able to stretch the clock.

I3C/I^{2}C target needs a clock stalling of up to 100 µs (at different phases)

If the delay requested by the target is less than 100 µs, MIPI® has standardized optional features for the I3C controller to stall-low the SCL clock up to 100 µs.

This I3C peripheral, when acting as controller, fully supports clock stalling and can be reconfigured without disabling the I3C controller, with:

- One programmable value of up to 100 µs, with a typical granularity of 1µs.
- Seven independent programmable enable bits, to possibly stall at the following I3C bus phases:
  - On I^{2}C legacy address ACK/NACK bit
  - On I^{2}C legacy write byte-level data ACK/NACK bit
  - On I^{2}C legacy read byte-level data byte ACK/NACK bit
  - On I3C address ACK/NACK bit
  - On I3C CCC command parity bit
  - On I3C write byte-level data parity bit
  - On I3C read byte-level data parity bit

Note: This I3C peripheral as controller allows more than 100 µs; the programmed delay is relative to the kernel clock frequency.

IBI with pending read: for read, if I3C target needs a clock stalling above 100 µs

- The I3C target can initiate an IBI with a pending read notification (optional MIPI feature) when it is ready to be read.
- This I3C peripheral, both as controller and target, supports IBI with pending read notification.
- The controller is aware of the IBI MDB support for pending read notification from a target, by sending the direct GETCAPS CCC format 1 (returned bit 6 of the GETCAP3 byte).
- The I3C target issues an IBI with MDB[7:5] = 3'b101 without any optional payload, to notify to the controller that it is ready to be read.
- The I3C controller must then issue a private read on the next transaction to this target.

Maximum read turnaround time: for read, if I3C target needs a clock stalling above 100 µs

- If the delay requested by the target is more than 100 µs, the controller can get the maximum read turnaround time by a direct GETMXDS CCC (format 2, 5 bytes).
- The maximum read turnaround time is returned as 24 bits, for a value between 0 and 16 seconds.
- The controller then inserts a STOP between a write (index) and a read.

45.3 I3C implementation

45.3.1 I3C instantiation

There is a single I3C instance in the device.

45.3.2 I3C wake-up from low-power mode(s)

The I3C peripheral can wake up the device from a low-power mode, as detailed in *Table 519*. For more details about the wake-up capabilities, refer to *Section 45.13*.

Table 519. I3C wake-up

Wake-up
From Stop mode with SVOS high

45.3.3 I3C FIFOs

The FIFOs are implemented as defined in *Table 520*.

Table 520. I3C FIFOs implementation

FIFO	Content	Unit	Size (in unit)	Used as controller/target (rationale)
C-FIFO	(32-bit) Control words	Word	2	Controller (a frame can be based on multiple control words; this is not the case as target)
S-FIFO	(32-bit) Status words			Controller (target: status only in register mode)
TX-FIFO	Transmitted data	Byte	8	Controller and target
RX-FIFO	Received data			

45.3.4 I3C triggers

This feature is not available in this product: no hardware trigger signal is connected as an input to the I3C peripheral.

45.3.5 I3C interrupt(s)

The interrupt mapping is implemented as detailed in *Table 521*.

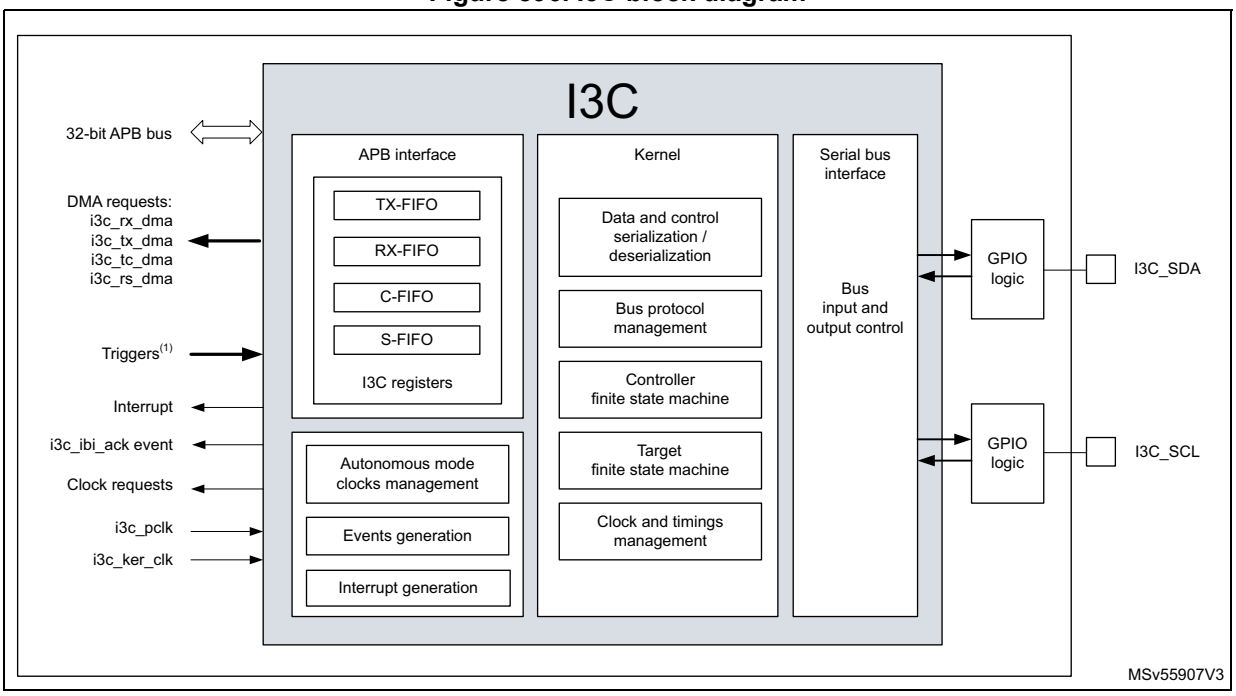
Table 521. I3C interrupt(s)

Signal name	Signal type	Description
i3c_err_it	O	Error interrupt line
i3c_evt_it	O	Event interrupt line

45.4 I3C block diagram

The I3C block diagram is illustrated in *Figure 896*.

Figure 896. I3C block diagram



1. Refer to *Section 45.3.4: I3C triggers*.

45.5 I3C pins and internal signals

Table 522. I3C input/output pins

Pin name	Pin type	Description
I3C_SDA	Input/output	Serial bus data line
I3C_SCL	Input/output	Serial bus clock line

Note: During initialization, the selected GPIOs for I3C_SCL and I3C_SDA should be configured as alternate function and no pull.

Table 523. I3C internal input/output signals

Signal name	Signal type	Description
i3c_pclk	I	APB clock
i3c_ker_clk	I	Kernel clock (also named as I3CCLK)
i3c_pclk_req	O	APB clock request
i3c_ker_clk_req	O	Kernel clock request
i3c_it(1)	O	Global interrupt line

Table 523. I3C internal input/output signals (continued)

Signal name	Signal type	Description
i3c_err_it(1)	O	Error interrupt line
i3c_evt_it(1)	O	Event interrupt line
i3c_rx_dma	O	DMA request for reading received bytes/words from RX-FIFO
i3c_tx_dma	O	DMA request for writing to be transmitted bytes/words to TX-FIFO
i3c_tc_dma	O	DMA request for writing to be transmitted control words to C-FIFO, when the I3C peripheral acts as controller
i3c_rs_dma	O	DMA request for reading status words from S-FIFO, when the I3C peripheral acts as controller
i3c_ibi_ack	O	IBI acknowledge event when the I3C peripheral acts as controller, for timing control with asynchronous mode 0

1. Refer to *Section 45.3.5: I3C interrupt(s)*.

45.6 I3C reset and clocks

45.6.1 I3C reset

On a system reset, the I3C peripheral is reset.

Alternatively, the software can reset specifically the I3C peripheral by writing the corresponding reset control bit (I3CxRST) of the reset and clock controller (RCC). Refer to the RCC section of this document for more details.

Additionally, when acting as target, the enabled I3C peripheral (EN = 1 in the I3C_CFGR register) can receive an in-band reset pattern on the I3C bus from the controller. The software is then notified (when RSTF = 1 in the I3C_EVR register and/or the corresponding interrupt is enabled) to perform the requested action, as registered in RSTACT[1:0] of the I3C_DEVR0 register, on the former reception of the broadcast or direct RSTACT CCC. Refer to *Table 527* and *Section 45.16.17* for more details.

This reset interrupt notification can be used to wake up from a low power mode.

For more details about the corresponding low-power mode(s), refer to the power management in the PWR section.

45.6.2 I3C clocks and requirements

As indicated in *Figure 896*, the I3C peripheral is implemented with several clock domains:

- SCL bus clock: for the I3C bus interface
  - When controller: the user must set and can adjust SCL/SDA timings by programming *I3C timing register 0 (I3C_TIMINGR0)*, *I3C timing register 1 (I3C_TIMINGR1)* and *I3C timing register 2 (I3C_TIMINGR2)*, as summarized in *Controller initialization* and *Updating the configuration for a transfer, as controller*.
  - When target: the user must set and comply with the bus available condition (t_{AVAL} for an in-band interrupt or controller-role request), and the bus idle condition (t_{IDLE} for a hot-join request), by programming *I3C timing register 1 (I3C_TIMINGR1)*, as summarized in *Target initialization*.

- I3CCLK kernel clock: for the I3C protocol management, data and control serialization/deserialization, controller and target finite state machines, bus clock and timings management
- APB clock: for the APB interface, DMA interface, events, and interrupt generation

APB clock and kernel clocks are driven from independently programmed clock sources via the RCC (refer to section Reset and clock control (RCC) for more details).

I3C kernel clock requirement, as controller

According to the intended value of the SCL clock on the bus, the application must guarantee that the frequency of the I3CCLK kernel clock be at least 2x the frequency of the SCL clock

Note: Sustaining $F_{SCL\ max}$ = 12.9 MHz means a frequency of the I3CCLK kernel clock > 25.8 MHz.

I3C kernel clock requirements, as target

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the I3CCLK kernel clock, meeting the following constraints:

1. Period of the I3CCLK kernel clock < t_{HIGH} (SCL clock high period)
   - $t_{HIGH\ min}$ = 24 ns. A frequency higher than 41.7 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
2. Period of the I3CCLK kernel clock < t_{CASr} (clock after repeated start condition)
   - $t_{CASr\ min}$ = $t_{CAS\ min}$ / 2 = 19.2 ns. A frequency higher than 52 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
3. Two periods of the I3CCLK kernel clock < t_{LOW_OD} (SCL clock low period in open drain)
   - $t_{LOW_OD\ min}$ = 200 ns. A frequency higher than 10 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
4. Frequency of the I3CCLK kernel clock > 2.5x frequency of the SCL clock
   - $F_{SCL\ max}$ = 12.9 MHz. A frequency higher than 32.3 MHz guarantees this constraint, which can be relaxed, depending on the I3C controller

Note: For a serial I3C/I^2C bus at a speed of Fm+ (up to 1 MHz), Fm (up to 400 KHz) or Sm (up to 100 kHz), I3C kernel clock must have a frequency respectively higher than 4 MHz, 2 MHz and 250 kHz.

APB clock requirement

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the APB clock:

APB clock period < 3x (SCL clock period) - I3CCLK kernel clock period

This means that $F_{APB} > [F_{SCL} \times F_{I3CCLK} / (3x\ F_{I3CCLK} - F_{SCL})]$

Note: This equation can be simplified to a minimum value of 5 MHz for the APB frequency.

For a serial I3C or Fm+ I^2C bus, a minimum APB clock of 400 kHz is enough.

45.7 I3C peripheral state and programming

45.7.1 I3C peripheral state

The I3C peripheral plays the role of I3C bus controller, or the role of an I3C target. In any case (see *Figure 897*), the peripheral is in one of the following states:

- Disabled state:
  - After an I3C reset (system reset or I3C reset from RCC), the I3C peripheral is in disabled state.
  - When the software sets to 1 bit EN in the I3C_CFGR register, the I3C peripheral takes into account the value of the different initialized configuration registers, and switches to the (enabled and) idle state. The peripheral is initialized as I3C target or I3C controller.
  - As I3C controller, then software can enable the I3C peripheral and initialize the I3C bus.
  - As I3C target, then the software can enable the I3C peripheral to operate on an I3C bus or an I^2C bus:

  a) If initialized as an I3C target without a programmed 7-bit static I^2C address, the software may issue a hot-join request on the I3C bus before being assigned a dynamic address by the I3C primary controller.

  b) If initialized as an I3C target with a programmed 7-bit static I^2C address, it can also automatically detect that it is connected to an I3C bus on the first I3C header sent by the I3C controller. Then it disables automatically both 20 mA open drain driver and spike filter to act as an I3C target, before being assigned a dynamic address.

  c) If initialized as an I3C target with a programmed 7-bit static I^2C address, it can also operate on an I^2C bus and respond to legacy I^2C messages, provided that the I^2C bus is restricted to the I^2C sub-features that are compliant with a MIPI I3C bus protocol: only Sm, Fm, or Fm+ speed, no clock stretching, no multiple 7-bit addresses, no general call message, no 10-bit addressing, no SMBus.
- Idle state:
  - After being enabled (EN = 1), the I3C peripheral activates its I3CCLK and SCL clock domains, and is able to communicate on the I3C bus.
  - The software can partly update the I3C peripheral configuration, see *Updating the configuration for a transfer, as controller* and *Updating the configuration of the I3C peripheral, as target* for more details.
  - The I3C peripheral switches to the (enabled and) active state in either one of the following conditions:

    Once the software initiates a transfer (as controller: when the software initiates a frame transfer; as target: when the software initiates an IBI/CR/HJ request).

    Once the hardware receives a request from another I3C device on the bus (as controller: after a start request from a target and a maximum T_{CAS} time; as target: when receiving a broadcast/direct CCC or a private read/write or a legacy I^2C message).
- Active state:
  - The I3C peripheral executes the transfer(s) on the bus.
  - When the requested transfer(s) is(are) completed, the software is notified by an event from the *I3C event register (I3C_EVR)*, and the corresponding interrupt is

enabled by *I3C interrupt enable register (I3C_IER)*. The I3C peripheral switches to idle state, is still able to communicate on the bus, and can be (partly) reconfigured:

As controller: the raised event/flag can be frame completed (FCF), IBI/controller-role/hot-join request completed (IBIF/CRF/HJF), or transfer error (ERRF).

As target: the raised event/flag can be dynamic address assignment completed (DAUPDF), IBI completed (IBIENDF), controller-role gaining completed (CRUPDF), broadcast/direct CCC completed (xxUPDF/RSTF/GETF/STAF), private read/write completed or legacy I^2C message completed (FCF), or transfer error (ERRF).

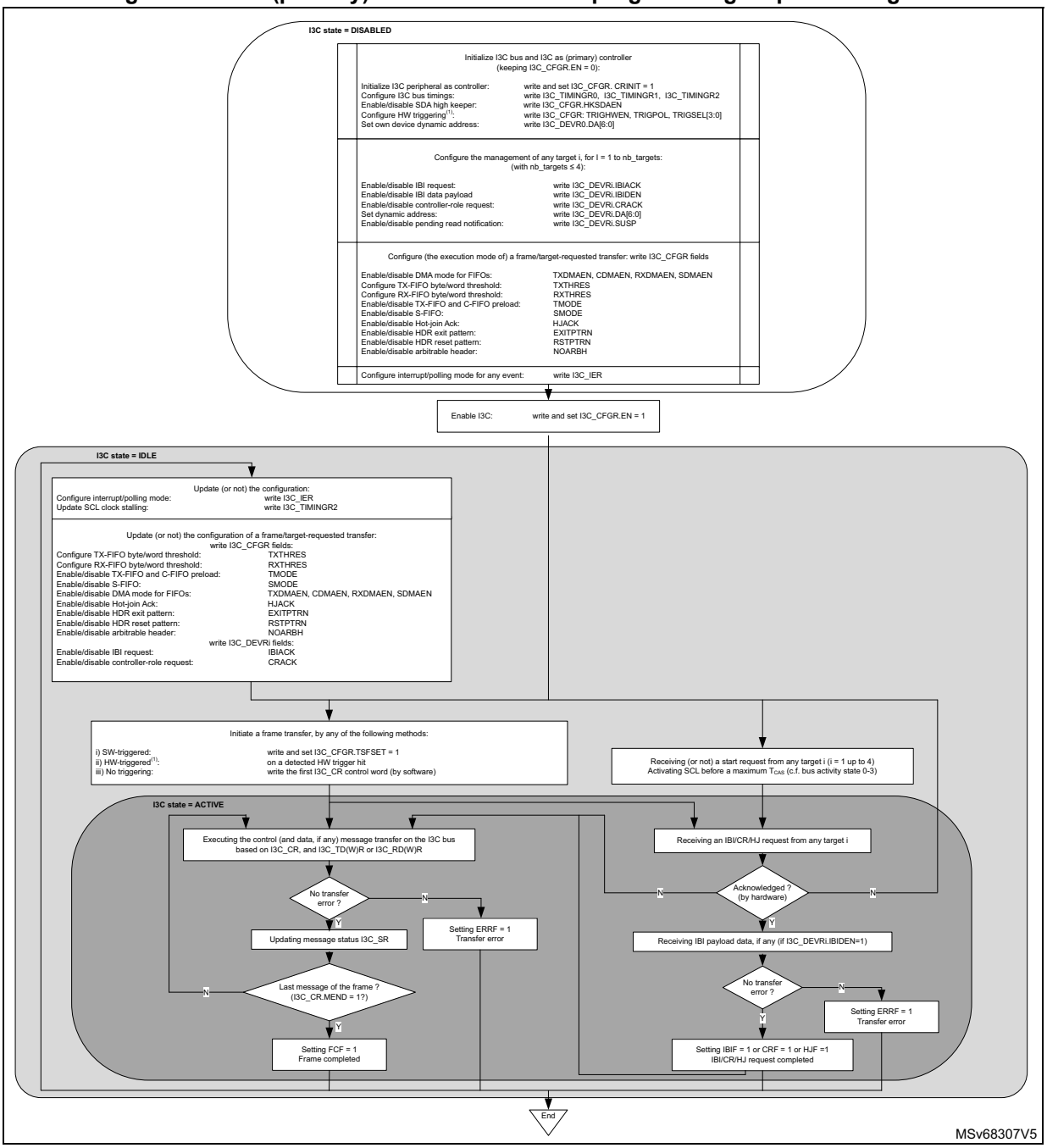
Note: The software can disable the I3C peripheral (write EN = 0), partially resetting it (subparts within the SCL clock domain and the I3CCLK kernel clock domain). Event, interrupt, and clock request generation are also impacted. The previously written configuration of the APB registers is kept and not modified.

Note: The software can disable the generation of the FCF flag by writing 1 into the FCFDIS bit of the I3C_CFGR register. This can be used with hardware triggers and autonomous DMA (if this feature is available, see Section 45.3), to stream a series of frames.

45.7.2 I3C controller state and programming sequence

Figure 897 illustrates the overall programming sequence of the I3C peripheral acting as (primary) controller, including state transitions, main subtasks, and conditions, as explained in this section.

Figure 897. I3C (primary) controller state and programming sequence diagram



1. This feature is implementation-dependent and can be unavailable. Refer to *Section 45.3.4: I3C triggers*.

Controller initialization

When the controller is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Configure *I3C configuration register (I3C_CFGR)* with the following fields:
  - CRINIT = 1: as I3C bus controller
  - HKSDAEN: high keeper on SDA enable/disable
- Configure I3C bus timings:

  a) *I3C timing register 0 (I3C_TIMINGR0)*:
  - SCL clock high time period (t_{DIG_H}, $t_{DIG_H_MIXED}$) in legacy I^2C and I3C open-drain/push-pull
  - SCL clock low time period (t_{DIG_L}, $t_{DIG_OD_L}$) in legacy I^2C and I3C open-drain/push-pull

  b) *I3C timing register 1 (I3C_TIMINGR1)*:
  - SDA hold time in push-pull (t_{HD_PP})
  - bus free condition time (I3C $t_{CAS,}$ legacy I^2C t_{BUF})
  - I3C repeated start timing (t_{CASr}, t_{CBSr})
  - I3C stop timing (t_{CBP})
  - SCL clock low maximum stalling on the ENTDAA CCC ($t_{STALLDAA}$), or on the ACK/NACK of a legacy I^2C or address phase of an I3C transfer, or the parity bit of a write data transfer, or on the ACK/NACK data phase of a legacy I^2C write, or on the transition bit of an I3C read transfer, or on the ACK/NACK phase of a legacy I^2C write (t_{STALL}), to adjust SCL clock low stalling, if needed by the I3C peripheral itself, when used as controller
  - $t_{NEWCRLock}$ for controller-role hand-off procedure (after GETACCCR CCC)

  c) *I3C timing register 2 (I3C_TIMINGR2)*:
  - SCL clock low stalling time, with separated enable/disable for each phase, to adjust SCL clock low stalling, if needed on the SDA hand-off with the addressed I3C target or legacy I^2C target
- Configure its own dynamic address: DA [6:0] field of the *I3C own device characteristics register (I3C_DEVR0)*
- Configure the management of any device target x: *I3C device x characteristics register (I3C_DEVRx)*, for x = 1 to x ≤ 4
- Configure the execution mode of a frame transfer or a target-requested transfer: *I3C configuration register (I3C_CFGR)*, with the following fields:
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN: DMA mode enable/disable for respectively, TX-FIFO, C-FIFO, RX-FIFO, S-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - TMODE: transmit mode (enable/disable for both TX-FIFO and C-FIFO preload)
  - SMODE: S-FIFO enable/disable
  - EXITPTRN, RSTPTRN: exit, reset pattern enable/disable
  - HJACK: hot-join acknowledge enable/disable
  - NOARBH: arbitrable header disable/enable
  - FCFDIS: FCF generation enable/disable
- Configure interrupt generation or polling mode from any event: *I3C interrupt enable register (I3C_IER)*

Then, the software can enable the I3C peripheral (set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR while enabling it.

Start a controller-initiated frame transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), the software can initiate a frame transfer by any of the following configuration methods:

- Software-triggering: on a write and set TSFSET = 1 in the I3C_CFGR register
  - This causes the hardware to raise the flag CFNFF = 1 in the I3C_EVR register, to request a first control word I3C_CR to be written.
- No triggering: on a write of the first control word I3C_CR by software.

Then, regardless of the frame starting method, the I3C peripheral switches to active state. While the control word is not the last message of the I3C frame (while MEND = 0 in the I3C_CR register), and while there is no transfer error (while ERRF = 1 in the I3C_EVR register), the hardware keeps requesting a next control word, and continuing the frame transfer:

- If the C-FIFO in not configured in DMA mode (CDMAEN = 0 in the I3C_CFGR register), the software writes a next control word following the flag CFNFF = 1 in the I3C_EVR register, or the corresponding interrupt if enabled (if CFNFIE = 1 in I3C_IER register)
- If the C-FIFO in configured in DMA mode (CDMAEN = 1), a next control word is automatically pushed and written by the allocated DMA channel consequently to the asserted I3C DMA request (i3c_tc_dma).

Start and receiving a target-initiated transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), concurrently to a possible controller-initiated transfer, a target can initiate a transfer by issuing a start request (drive SDA low), provided the controller has allowed a hot-join request, an IBI request, or a controller-role request via the I3C_DEVR0 register.

In this case, even though the controller software has no intent to start a frame transfer, the hardware switches to active state (activates the SCL clock before a maximum t_{CAS} time defined as 1 µs, 100 µs, 2 ms, or 50 ms, depending on, respectively, the bus activity state 0, 1, 2, or 3) to receive the hot-join/in-band interrupt/controller-role request from the target.

For more information about the execution of target-initiated I3C bus transfer and its related programming as a controller, refer to the relevant figures in *Section 45.9*:

- *Figure 912: IBI transfer, as controller/target*
- *Figure 913: Hot-join request transfer, as controller/target*
- *Figure 914: Controller-role request transfer, as controller/target*

Executing a (controller-initiated) frame transfer

The controller executes on the bus the frame transfer until the completion of the last message (FCF = 1 in the I3C_EVR register), or a transfer error (ERRF = 1 in the I3C_EVR register), and the corresponding interrupt, if enabled. This is based on I3C_CR and I3C_TD(W)R registers, written explicitly by software or pushed by the allocated DMA channel, and based on the I3C_RD(W)R, read explicitly by the software or by the allocated DMA channel. Then the I3C controller switches back to idle state.

For more information about the execution of controller-initiated I3C bus transfer and its related programming as a controller, refer to figures in *Section 45.9*:

- *Figure 901: I3C CCC messages, as controller*
- *Figure 902: I3C broadcast ENTDAA CCC, as controller*
- *Figure 903: I3C broadcast, direct read and direct write RSTACT CCC, as controller*
- *Figure 908: I3C private read/write messages, as controller*
- *Figure 910: Legacy I2C read/write messages, as controller*

Figure 897 does not include the management of the FIFOs (TX-FIFO, RX-FIFO, C-FIFO, and S-FIFO). This is detailed in *Section 45.10*.

For each completed message without transfer error, the hardware reports the exchanged transfer on the I3C bus by updating *I3C status register (I3C_SR)*, which can be read or not by the software when the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register).

- In the case of a direct CCC read or a private read transfer, in addition to the completion of the last message (FCF = 1 in the I3C_EVR register) or a transfer error (ERRF = 1 in the I3C_EVR register) and the corresponding interrupt if enabled, and provided that the S-FIFO is disabled for the status register I3C_SR (SMODE = 0 in the I3C_CFGR register), the software is notified if the read transfer is ended prematurely by the target by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt, if enabled. The software can then read I3C_SR, to get more information about the executed transfer.

Alternatively, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the status register I3C_SR must be read for each executed message, either directly by the software (notified by SFNEF = 1 in the I3C_EVR register and the corresponding interrupt, if enabled), or via the DMA (if SDMAEN = 1 in the I3C_CFGR register), no matter if a read is prematurely ended by the target or not. Frame completion (FCF = 1 in the I3C_EVR register) occurs only after reading the status of the last message (S-FIFO is empty). For more information, refer to *Section 45.10.4*.

Updating the configuration for a transfer, as controller

Back in idle state, the software can update the configuration of the I3C peripheral before the next transfer:

- Modify SCL clock stalling via *I3C timing register 2 (I3C_TIMINGR2)*
- Modify the interrupt/polling mode policy via *I3C interrupt enable register (I3C_IER)*
- Modify the following fields of the *I3C configuration register (I3C_CFGR)*:
  - TXTHRES, RXTHRES
  - TMODE, SMODE
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN
  - EXITPTRN, RSTPTRN
  - NOARBH
  - FCFDIS

- Modify/prepare the control words, status words, read/write data of the next frame transfer to be executed, by software and/or DMA
  - *I3C message control register (I3C_CR)*, *I3C message control register [alternate] (I3C_CR)*
  - *I3C status register (I3C_SR)*
  - *I3C transmit data byte register (I3C_TDR)*, *I3C transmit data word register (I3C_TDWR)*
  - *I3C receive data byte register (I3C_RDR)*, *I3C receive data word register (I3C_RDWR)*
- Typically after having issued and completed a broadcast/direct DISEC/ENEC CCC:
  - Modify the hot-join acknowledge policy via bit HJACK in the I3C_CFGR register
  - Modify IBI/CR acknowledge policy to any target x, via *I3C device x characteristics register (I3C_DEVRx)*

The registers usage versus the I3C peripheral role as controller is summarized in *Section 45.8.1*.

The static/dynamic registers fields usage when acting as controller is summarized in *Table 525*.

45.7.3 I3C target state and programming sequence

The application software can initialize the I3C peripheral as I3C target to operate either:

a) **on an I3C bus without a programmed 7-bit static I²C address (if I3C_I2CCFGR.SAVAL = 0 and I3C_I2CCFGR.MAVAL = 0):** the software may issue a hot-join request on the I3C bus before being assigned a dynamic address by the I3C primary controller via the broadcast ENTDAA CCC. This is the typical I3C target use case.

b) **on an I3C bus with a programmed 7-bit static I²C address (if I3C_I2CCFGR.SAVAL = 1 or I3C_I2CCFGR.MAVAL = 1):** the peripheral can automatically detect that it is connected to an I3C bus on the first I3C header sent by the I3C controller. Then it disables automatically both 20 mA open drain driver and spike filter to act as an I3C target, before being assigned a dynamic address via the broadcast SETAASA or the direct SETDASA. Note: ENTDAA does not need the target to be initialized with a static address; but ENTDAA may be also used and acknowledged by the target also in this b) case, provided that the target has not received SETAASA nor SETDASA earlier. If SETDASA or SETAASA is first received and completed, I3C_DEVR0.DAVAL is set to 1 by the hardware and the target will not participate into a next ENTDAA before a RSTDAA or RSTACT. Afterwards, the peripheral can operate as a typical I3C target with a dynamic address, possibly issue in-band interrupts and respond to I3C messages.

c) **on an I²C bus with a programmed 7-bit static I²C address (if I3C_I2CCFGR.SAVAL = 1):** the peripheral can respond to legacy I²C messages, provided that the I²C bus is restricted to the I²C sub-features that are compliant with a MIPI I3C bus protocol: only Sm, Fm, or Fm+ speed, no clock stretching, no multiple 7-bit addresses, no general call message, no 10-bit addressing, no SMBus.

Figure 898, *Figure 899*, and *Figure 900* illustrate the overall programming sequence of the I3C peripheral acting as target, including state transitions, main subtasks, and conditions, as explained in this section, for each of these use cases.

Figure 898. I3C target state and programing sequence diagram, when initialized on an I3C bus without a static address

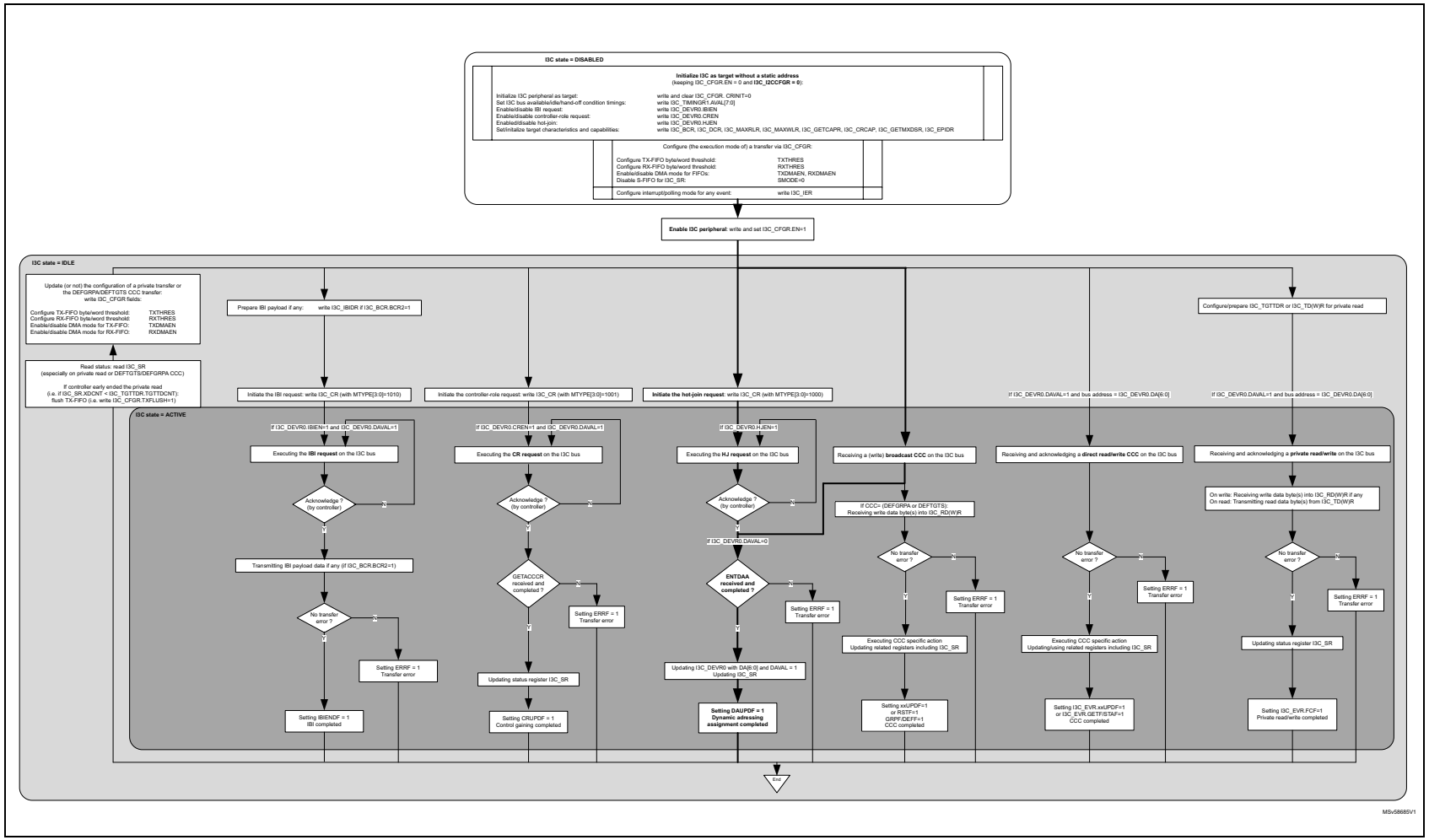


Figure 899. I3C target state and programing sequence diagram when initialized on an I3C bus with a static address,

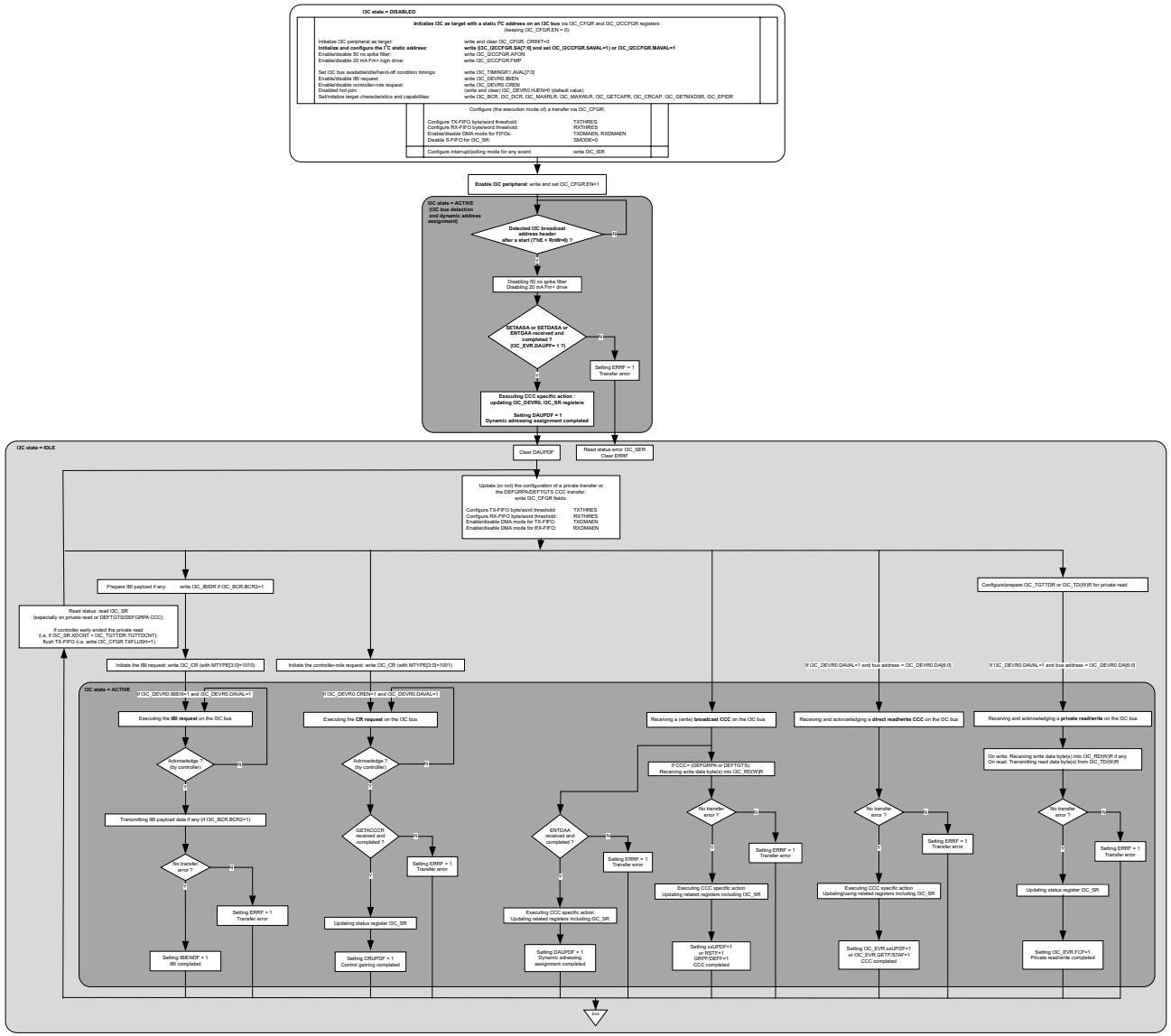
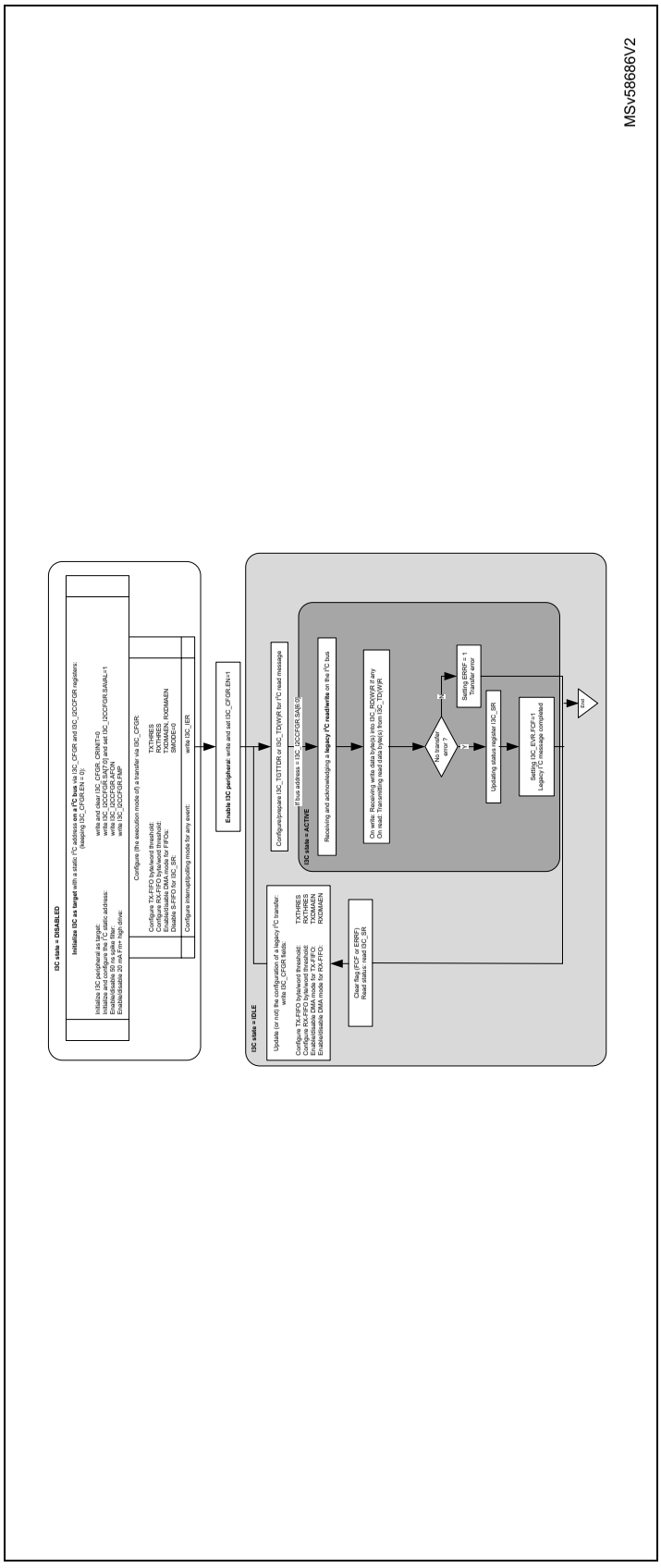


Figure 900. I3C target state and programing sequence diagram, when initialized on an I²C bus with a static address



Target initialization

When the target is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Set the I3C peripheral as target: write and clear CRINIT = 0 in the I3C_CFGR register
  a) Initialize the I3C peripheral on an I3C bus without a 7-bit address: write *I3C I2C target configuration register (I3C_I2CCFGR)* with:
  – Cleared SAVAL = 0 and MAVAL = 0
  – Cleared AFON = 0
  – Cleared FMP = 0
  b) Initialize the I3C peripheral as target on an I3C bus with a 7-bit address: write *I3C I2C target configuration register (I3C_I2CCFGR)* with:
  – Static address value: (write SA[6:0] and set SAVAL = 1) or set MAVAL = 1
  – Cleared AFON = 0
  – Cleared FMP = 0
  c) Initialize the I3C peripheral as target on an I2C bus: write *I3C I2C target configuration register (I3C_I2CCFGR)* with:
  – Static address value: write SA[6:0] and set SAVAL = 1
  – Enable (for Fm or Fm+) or disable the 50 ns spike filter: AFON
  – Enable (for Fm+) or disable the 20 mA high drive: FMP
- If a) or b): set the I3C bus timings via *I3C timing register 1 (I3C_TIMINGR1)*: write AVAL[7:0] for:
  – Bus available condition time (t_{AVAL}) for IBI or controller-role request
  – Bus idle condition time (t_{IDLE}) for hot-join request
  – $t_{NEWCRLock}$ for controller-role hand-off procedure (after GETACCCR CCC)
- If a) or b): configure target-initiated requests: write *I3C own device characteristics register (I3C_DEVR0)* with
  – IBIEN: in-band interrupt (also known as IBI) request enable/disable
  – CREN: controller-role request enable/disable
  – HJEN: hot-join request enable/disable
- If a) or b): initialize target characteristics and capabilities: write
  – *I3C bus characteristics register (I3C_BCR)*
  – *I3C device characteristics register (I3C_DCR)*
  – *I3C maximum read length register (I3C_MAXRLR)*
  – *I3C maximum write length register (I3C_MAXWLR)*
  – *I3C get capability register (I3C_GETCAPR)*
  – *I3C controller-role capability register (I3C_CRCAPR)*
  – *I3C get max data speed register (I3C_GETMXDSR)*
  – *I3C extended provisioned ID register (I3C_EPIDR)*
- Configure the execution mode of a transfer: *I3C configuration register (I3C_CFGR)*, with the following fields:
  – TXDMAEN, RXDMAEN: DMA mode enable/disable for, respectively, TX-FIFO and RX-FIFO
  – TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold

  - Disable S-FIFO: SMODE = 0 (default/reset value)
- Configure interrupt generation or polling mode from any event: *I3C interrupt enable register (I3C_IER)*

Then, the software can enable the I3C peripheral (write and set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR register while enabling it.

Receiving a (broadcast CCC, direct read/write CCC or private read/write) message on the I3C bus

When the target is in idle state (EN = 1 in the I3C_CFGR register), the target is ready to receive a communication message on the I3C bus from the controller, and is ready to switch to the active state.

Typically, first, the active target is receiving a broadcast ENTDAA CCC, possibly after optional received broadcast ENEC/DISEC CCC(s), and is then assigned a dynamic address. The event DAUPF in the I3C_EVR register is raised to 1, the related interrupt is generated if enabled, and the target goes back to idle state.

After that, the idle target is ready to receive any other broadcast CCC message, or direct read/write CCC, or private read/write message from the controller.

For more information about the execution of controller-initiated I3C bus transfers and its related programming as a target, including the updated I3C registers and fields, refer to figures in *Section 45.9*:

- *Figure 904: I3C CCC messages, as target*
- *Figure 905: I3C broadcast ENTDAA CCC, as target*
- *Figure 906: I3C broadcast DEFTGTS CCC, as target*
- *Figure 907: I3C broadcast DEFGRPA CCC, as target*
- *Figure 909: I3C private read/write messages, as target*

Figure 898, *Figure 899* and *Figure 900* do not include the FIFOs management (TX-FIFO, RX-FIFO). This is detailed in *Section 45.10*.

Receiving a legacy I^2C read/write message on the I^2C bus

When the target is in idle state (EN = 1 in the I3C_CFGR register), the target is ready to receive a legacy I^2C read/write message from the controller.

For more information about the execution of controller-initiated I^2C bus transfers and its related programming as a target, including the updated I3C registers and fields, refer to figures in *Section 45.9*:

- *Figure 911: Legacy I^2C read/write messages, as target*

Figure 900 does not include the FIFOs management (TX-FIFO, RX-FIFO). This is detailed in *Section 45.10*.

Read the message status register

For each received and completed message without transfer error, the hardware reports the exchanged transfer on the bus by updating the *I3C status register (I3C_SR)*, which can be read by the software after being notified by the corresponding flag in the *I3C event register (I3C_EVR)*, or by the corresponding interrupt if enabled in the *I3C interrupt enable register (I3C_IER)*.

I3C status register (I3C_SR) must be read by the software after the following messages:

- A private read or a legacy I^2C read: to get the number of exchanged data bytes, as the controller can have ended the transfer earlier than expected by the target (if XDCNT[15:0] in the I3C_SR register is lower than TGTTDCNT[15:0] in the I3C_TGTTDR register). If so, software must flush the TX-FIFO (write TXFLUSH = 1 in the I3C_CFGR register).
- A DEFTGTS CCC or a DEFGRPA CCC: to get the number of received data bytes in the RX-FIFO

Start a (target-initiated) transfer on the I3C bus

When the target goes first from disabled to idle state (software writes EN = 1 in the I3C_CFGR register), concurrently to be able to receive a broadcast CCC from the controller, the software can initiate a hot-join request (the software writes MTYPE[3:0] = 1000 in the I3C_CR register) to be eligible to participate to a next ENTDAA CCC, provided it is allowed to do so (HJEN = 1 in the I3C_DEVR0 register).

Once a dynamic address is assigned (DAUPF = 1 in the I3C_EVR register), more generally and possibly concurrently to a frame transfer emitted by the controller, the software can initiate an IBI (in-band interrupt request) to the controller, or a controller-role request by writing the related control word into the I3C_CR register.

For more information about the execution of target-initiated I3C bus transfer, and its related programming as a target, refer to figures in *Section 45.9*:

- *Figure 912: IBI transfer, as controller/target*
- *Figure 913: Hot-join request transfer, as controller/target*
- *Figure 914: Controller-role request transfer, as controller/target*

Updating the configuration of the I3C peripheral, as target

Back in idle state, the software can update the configuration of the I3C target before a next transfer:

- Modify the interrupt/polling mode policy via *I3C interrupt enable register (I3C_IER)*
- Modify following fields of the *I3C configuration register (I3C_CFGR)*:
  - TXTHRES, RXTHRES
  - TXDMAEN, RXDMAEN
- Modify/prepare the *I3C IBI payload data register (I3C_IBIDR)*, if any payload (if BCR2 = 1 in the I3C_BCR register), before initiating an IBI transfer (write *I3C message control register [alternate] (I3C_CR)* with MTYPE[3:0] = 1010)
- Modify/prepare *I3C target transmit configuration register (I3C_TGTTDR)*, to disable or enable the TX-FIFO to be preloaded with a defined number of data bytes to be transmitted, before receiving a private read or a direct CCC read (out of the GETSTATUS CCC) or a legacy I^2C read from the controller

The registers usage vs. the I3C peripheral role as target is summarized in *Section 45.8.1*.

The static/dynamic registers fields usage, when acting as target, is summarized in *Table 526*.

45.8 I3C registers and programming

45.8.1 I3C register set, as controller/target

Table 524 lists the registers and their usage versus the I3C peripheral role.

Table 524. I3C register usage

Register	Used as controller	Used as target
I3C_CR	X	X
I3C_CFGR	X	X
I3C_I2CCFGR	-	X
I3C_RDR	X	X
I3C_RDWR	X	X
I3C_TDR	X	X
I3C_TDWR	X	X
I3C_IBIDR	X	X
I3C_TGTTDR	-	X
I3C_SR	X	X
I3C_SER	X	X
I3C_RMR	X	X
I3C_EVR	X	X
I3C_IER	X	X
I3C_CEVR	X	X
I3C_MISR	X	X
I3C_DEVR0	X	X
I3C_DEVRx (x = 1 to 4)	X	-
I3C_MAXRLR	-	X
I3C_MAXWLR	-	X
I3C_TIMINGR0	X	-
I3C_TIMINGR1	X	X
I3C_TIMINGR2	X	-
I3C_BCR	-	X
I3C_DCR	-	X
I3C_GETCAPR	-	X
I3C_CRCAPR	-	X
I3C_GETMXDSR	-	X
I3C_EPIDR	-	X

45.8.2 I3C registers and fields use versus peripheral state, as controller

When the I3C peripheral acts as controller, *Table 525* lists the registers and their usage versus the controller state (disabled, idle, and active).

Table 525. I3C registers/fields usage versus controller state

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CR	X	-	-	X
I3C_CFGR	X	CRINIT HKSDAEN EN(1)	Write: any used field except {CRINIT, HKSDAEN}, namely {CDMAEN SDMAEN TXDMAEN RXDMAEN TMODE SMODE TXTHRES RXTHRES HJACK EXITPTRN RSTPTRN NOARBH}(2) FCFDIS CFLUSH SFLUSH TXFLUSH RXFLUSH TSFSET	-
I3C_RDR	X	-	-	Read
I3C_RDWR	X	-	-	Read
I3C_TDR	X	-	-	Write
I3C_TDWR	X	-	-	Write
I3C_IBIDR	X	-	-	Read
I3C_TGTTDR	-			
I3C_SR	X	-	-	Read
I3C_SER	X	-	Read	-
I3C_RMR	X	-	-	Read RADD[6:0] IBIRDCNT[2:0]

Table 525. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_EVR	X	-	-	Read (controller-role fields): HJF CRF IBIF FCF ERRF RXTGTENDF RXFNEF TXFNFF SFNEF CFNFF RXLASTF TXLASTF TXFEF CFEF
I3C_IER	X	Write xIE with x = HJ, CR, IBI, FC, ERR, RXTGTEND, RXFNE, TXFNF, SFNE, CFNF(2)		
I3C_CEVR	X	-	-	Write (controller-role fields, refer to I3C_EVR)
I3C_MISR	X	-	-	Read (controller-role fields): HJMIS CRMIS IBIMIS FCMIS ERRMIS RXTGTENDMIS RXFNEMIS TXFNFMIS SFNEMIS CFNFMIS
I3C_DEVR0	X	Write DA[6:0](2)		-
I3C_DEVRx x = 1..4	X	Write SUSP, IBIDEN, IBIACK, CRACK, DA[6:0](2)		-
I3C_MAXRLR	-			
I3C_MAXWLR	-			
I3C_TIMINGR0	X	X	-	-
I3C_TIMINGR1	X	X	-	-
I3C_TIMINGR2	X	Write(2)		
I3C_BCR	-			
I3C_DCR	-			
I3C_GETCAPR	-			
I3C_CRCAPR	-			

Table 525. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_GETMXDSR	-			
I3C_EPIDR	-			

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.
2. These fields are typically written and initialized in disabled state during bus configuration. They are not write-protected when EN = 0, and can be also written and updated in other state(s).

45.8.3 I3C registers and fields usage versus peripheral state, as target

When the I3C peripheral acts as target, *Table 526* lists the registers and their usage versus the I3C target state (disabled, idle, and active).

Table 526. I3C registers/fields usage versus target state

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CR	X	-	MTYPE[3:0] DCNT[2:0]	-
I3C_CFGR	X	CRINIT EN(1)	Write: any used field except CRINIT, namely: {TXDMAEN RXDMAEN TXTHRES RXTHRES}(2) TXFLUSH RXFLUSH	-
I3C_I2CCFGR	X	X (any field)	-	-
I3C_RDR	X	-	-	Read
I3C_RDWR	X	-	-	Read
I3C_TDR	X	-	-	Write
I3C_TDWR	X	-	-	Write
I3C_IBIDR	X	-	Write	-
I3C_TGTTDR	X	-	Write	-
I3C_SR	X	-	-	Read DIR, XDCNT[15:0]
I3C_SER	X	-	Read DOVR, STALL, PERR, CODERR[3:0]	-
I3C_RMR	X	-	-	Read RCODE[7:0]

Table 526. I3C registers/fields usage versus target state (continued)

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_EVR	X	-	-	Read (target-role fields): GRPF DEFF INTUPDF ASUPDF RSTF MRLUPDF MWLUPDF DAUPDF STAF GETF WKPF CRUPDF IBIENDF ERRF FCF RXFNEF TXFNFF TXLASTF TXFEF
I3C_IER	X	Write xIE with x = GRP, DEF, INTUPD, ASUPD, RST, MRLUPD, MWLUPD, DAUPD, STA, GET, WKP, CRUPD, IBIEND, ERR, FC, RXFNE, TXFNF(2)		
I3C_CEVR	X	-	-	Write (target-role fields, refer to I3C_EVR)
I3C_MISR	X			Read (target-role fields): GRPMIS DEFMIS INTUPDMIS ASUPDMIS RSTMIS MRLUPDMIS MWLUPDMIS DAUPDMIS STAMIS GETMIS WKPMIS CRUPDMIS IBIENDMIS ERRMIS FCMIS RXFNEMIS TXFNFMIS

Table 526. I3C registers/fields usage versus target state (continued)

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_DEVR0	X	HJEN CREN IBIEN	Read RSTVAL, RSTACT[1:0], and AS[1:0]	-
I3C_DEVRx x = 1..4	-			
I3C_MAXRLR	X	X	-	-
I3C_MAXWLR	X	X	-	-
I3C_TIMINGR0	-			
I3C_TIMINGR1	X	AVAL[7:0]	-	-
I3C_TIMINGR2	-			
I3C_BCR	X	X	-	-
I3C_DCR	X	X	-	-
I3C_GETCAPR	X	X	-	-
I3C_CRCAPR	X	X	-	-
I3C_GETMXDSR	X	X	-	-
I3C_EPIDR	X	X	-	-

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.
2. These fields are typically written and initialized in disabled state during I3C bus configuration. They are not write-protected when EN = 0.

45.9 I3C bus transfers and programming

45.9.1 I3C command set (CCCs), as controller/target

The list of the supported I3C command set (for example, list of CCCs, common command codes) and the overview of how they are handled by the I3C peripheral acting as controller or target, is specified in *Table 527*.

Table 527. List of supported I3C CCCs, as controller/target

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
Broadcast CCCs							
ENEC	0x00	Write	No defining/sub-command byte	With one data byte (enable target events byte)	X	X, INTUPDF	Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any
DISEC	0x01			With one data byte (disable target events byte)	X	X, INTUPDF	Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any
ENTASx x = 0...3	0x02 ... 0x05			No data byte	X	X, ASUPDF	Update I3C_DEVR0.AS[1:0]
RSTDAA	0x06			-	X	X, DAUPDF	Clear I3C_DEVR0.DAVAL = 0
ENTDAA	0x07			-	X	X, DAUPDF if DAVAL = 0	Update I3C_DEVR0: DA[6:0] and set DAVAL = 1
DEFTGTS	0x08			With [1+ 4x (1+ number_of_targets)] x data bytes	X	X, DEFF	Update I3C_RDR/ I3C_RDWR. Refer to *Figure 906*.
SETMWL	0x09			With two data byte	X	X, MWLUPDF	Update I3C_MAXWLR
SETMRL	0x0A			With 2 or 3 data bytes	X	X, MRLUPDF	Update I3C_MAXRLR
ENTTM	0x0B			With one data byte	X	-	-
SETXTIME	0x28		With sub-command byte	Without or with one or more data bytes	X	-	-
SETAASA	0x29		No defining/sub-command byte	No data byte	X	X, DAUPDF if SAVAL = 1 and DAVAL = 0	Update I3C_DEVR0 with DA[6:0] and DAVAL = 1
RSTACT	0x2A		With defining byte (0x00, 0x01 or 0x02)		X	X, RSTF after detected reset pattern	Update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1
DEFGRPA	0x2B		No defining/sub-command byte	With several data bytes	X	X, GRPF	Update I3C_RDR/ RDWR. Refer to *Figure 907*.
RSTGRPA	0x2C			No data byte	X	-	-

Table 527. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
Direct CCCs							Action if ACK (if I3C target Address = I3C_DEVR0.DA[6:0] and I3C_DEVR0.DAVAL = 1) (else NACK)
ENEC	0x80	Write	No defining/sub-command byte	With one data byte (enable target events byte)	X	X, INTUPDF	Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any
DISEC	0x81			With one data byte (disable target events byte)	X	X, INTUPDF	Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any
ENTASx x = 0...3	0x82...0x85			No data byte	X	X, ASUPDF	Update I3C_DEVR0.AS[1:0]
SETDASA	0x87			No data byte	X	X, DAUPDF if [(SAVAL=1 or MAVAL=1) and DAVAL=0]	Update I3C_DEVR0 with DA[6:0] and DAVAL = 1
SETNEWDA	0x88			With one data byte	X	X, DAUPDF if DAVAL = 1	Update I3C_DEVR0: DA[6:0] (and keep DAVAL = 1)
SETMWL	0x89			With two data bytes	X	X, MWLUPDF	Update I3C_MAXWLR
SETMRL	0x8A			With two or three data bytes	X	X, MRLUPDF	Update I3C_MAXRLR
GETMWL	0x8B	Read	No defining/sub-command byte	With two data bytes	X	X, GETF	Return data bytes from I3C_MAXWLR[15:0]. Refer to *Section 45.16.20*.
GETMRL	0x8C			With two or three data bytes	X	X, GETF	Return data bytes from I3C_MAXRLR[15:0] and if I3C_BCR.BCR2 = 1 return third byte from I3C_MAXRLR.IBIP[2:0]. Refer to *Section 45.16.19*.

Table 527. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
GETPID	0x8D	Read	No defining/sub-command byte	With six data bytes	X	X, GETF	Return data bytes from I3C_EPIDR. Refer to *Section 45.16.30*.
GETBCR	0x8E			With one data byte	X	X, GETF	Return data byte from I3C_BCR[7:0]. Refer to *Section 45.16.25*.
GETDCR	0x8F				X	X, GETF	Return I3C_DCR[7:0]. Refer to *Section 45.16.26*.
GETSTATUS	0x90		With or without defining byte (TGTSTAT, PRECR)	With two data bytes (format 1 or format 2 with PRECR)	X	X, STAF if format 1 X, GETF if format 2	Return 2 data bytes, as detailed in *Section 45.9.9*.
GETACCCR	0x91		No defining/sub-command byte	With one data byte	X	X, CRUPDF	Return data byte from I3C_DEVR0.DA[6:0] with parity bit
GETMXDS	0x94		With or without defining byte (WRRDTURN, CRHDLY)	With two data bytes (format 1) or 5 data bytes (format 2 or format 3 with WRRDTURN) or 1 data byte (format 3 with CRHDLY)	X	X, GETF	Return data byte(s) from I3C_GETMXDSR. Refer to *Section 45.16.29*.
GETCAPS	0x95		With or without defining byte (TGTSTAT, CRCAPS)	With 3 data bytes (format 1 or format 2 with TGTSTAT) or two data bytes (format 2 with CRCAPS)	X	X, GETF	Return 3 GETCAPx data bytes from I3C_GETCAPR (refer to *Section 45.16.27*) or Return 2 CRCAPx data bytes from I3C_CRCAPR (refer to *Section 45.16.28*)
D2DXFER	0x97	Write	With defining byte	With defining byte	X	-	-
SETXTIME	0x98		With sub-command byte	With sub-command byte	X		
GETXTIME	0x99	Read	No defining/sub-command byte	No defining/sub-command byte	X		

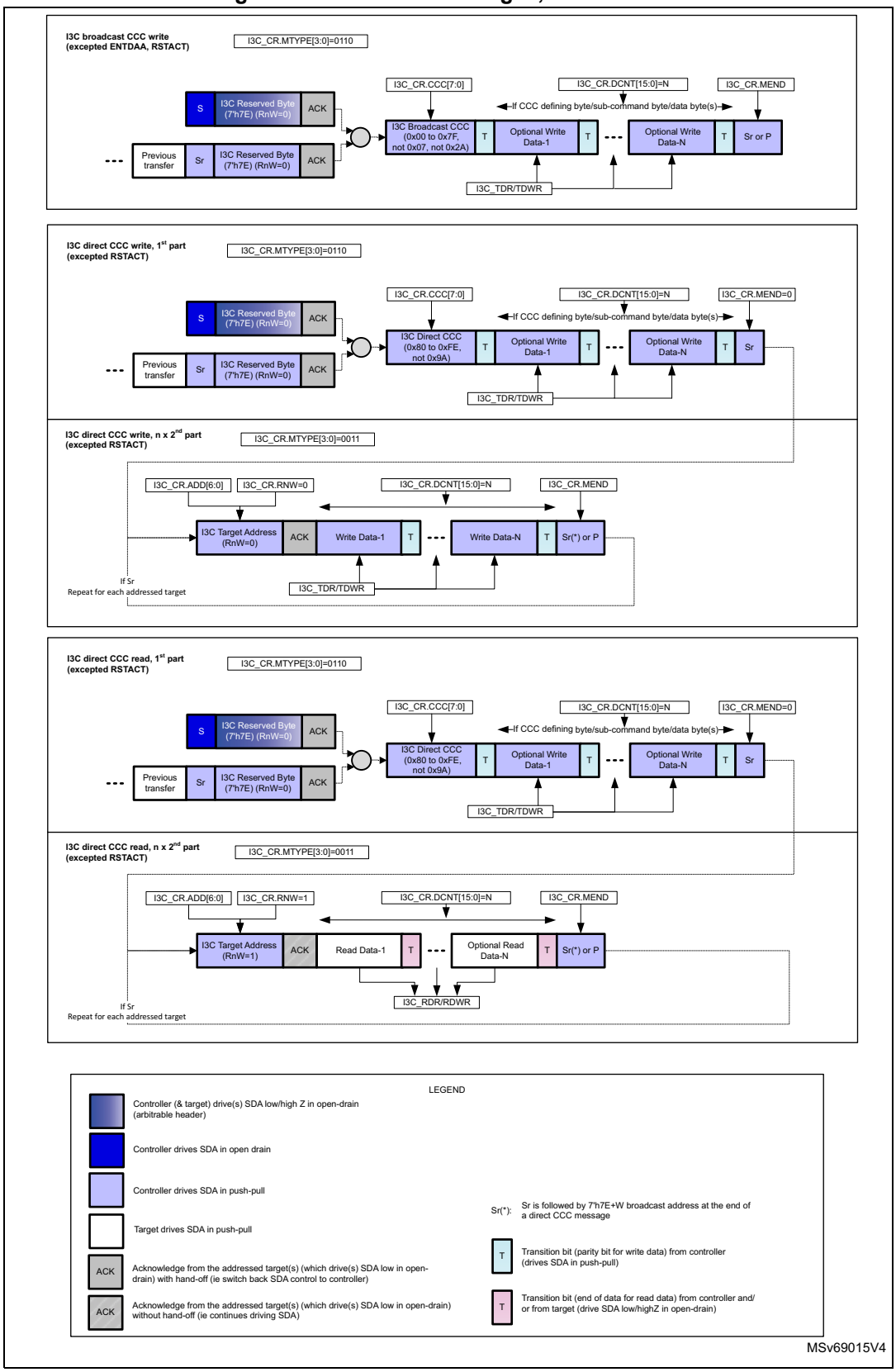
Table 527. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
RSTACT	0x9A	Read/ Write	With defining byte (0x00, 0x01, or 0x02)	With defining byte (0x00, 0x01, or 0x02)	X	X, RSTF if detected reset pattern	Read: return data byte from RSTACT[1:0] in the I3C_DEVR0 register. Write: update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1
SETGRPA	0x9B	Write	No defining/sub-command byte	No defining/sub-command byte	X	-	-
RSTGRPA	0x9C				X		

45.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller

Figure 901 illustrates I3C broadcast CCC write transfer (except ENTDAA, RSTACT), and direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

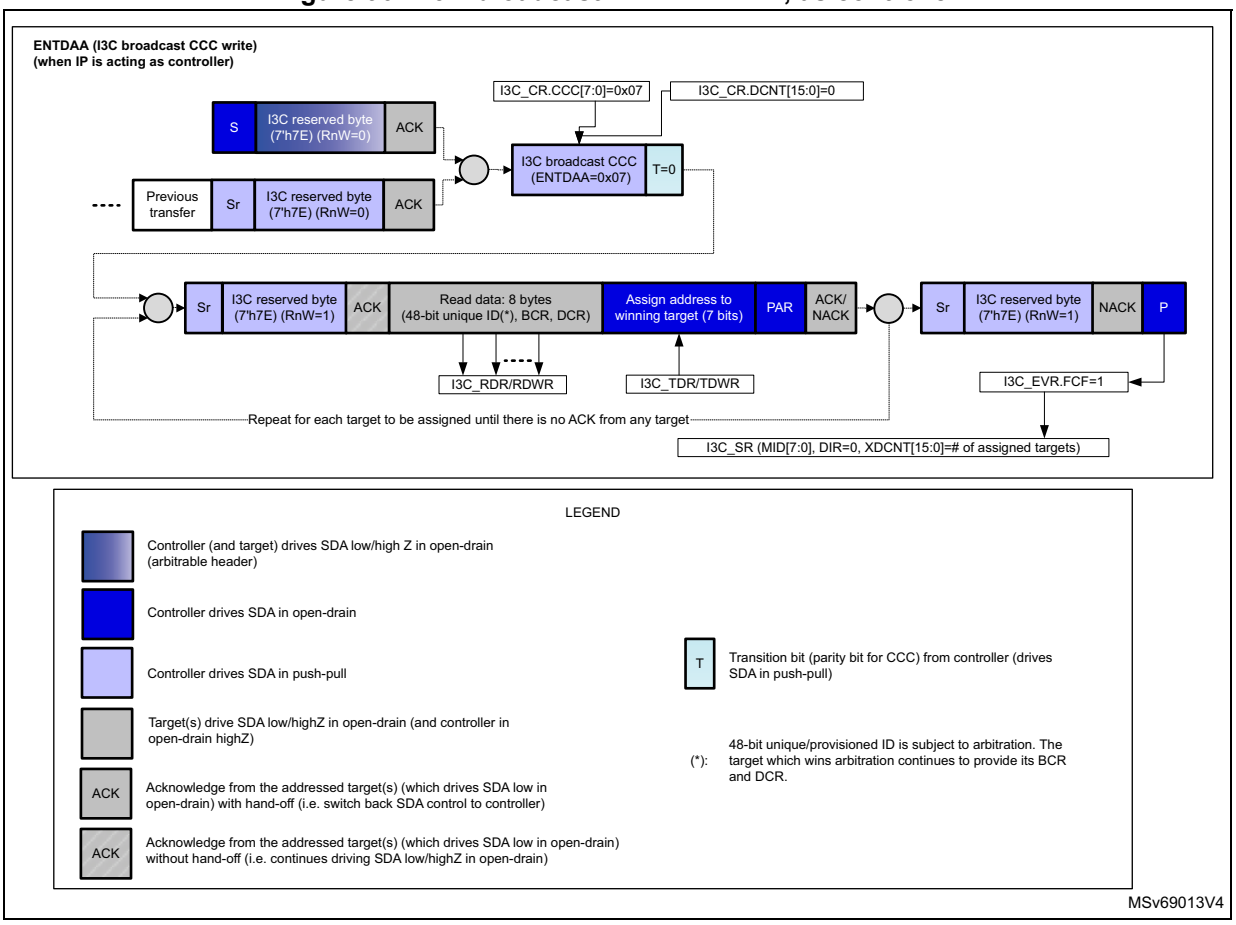
Figure 901. I3C CCC messages, as controller



45.9.3 I3C broadcast ENTDAA CCC transfer, as controller

Figure 902 illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as controller.

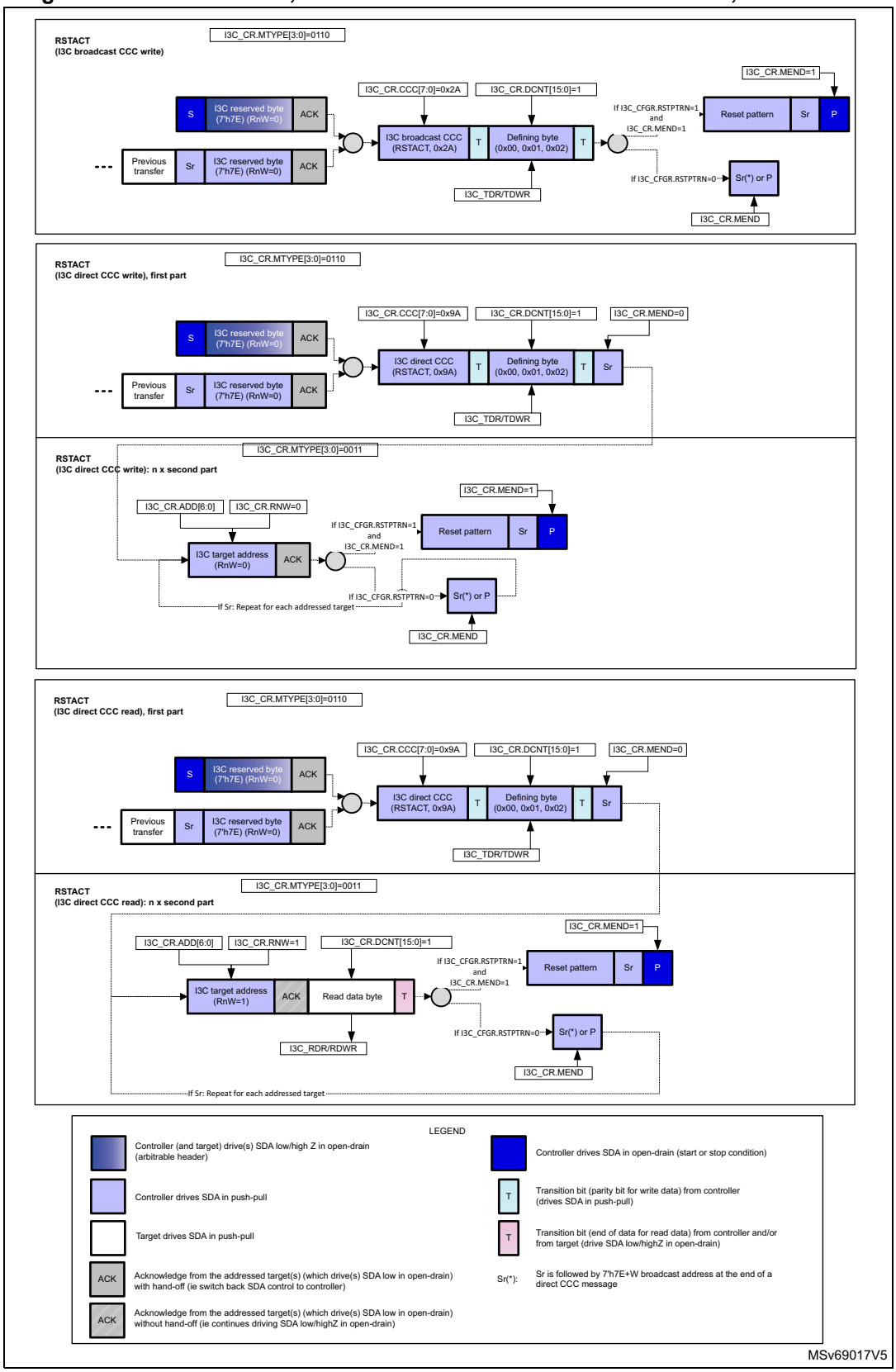
Figure 902. I3C broadcast ENTDAA CCC, as controller



45.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller

Figure 903 illustrates I3C broadcast (write), direct write and read RSTACT CCC, as communicated on the I3C bus, and as programmed when acting as controller.

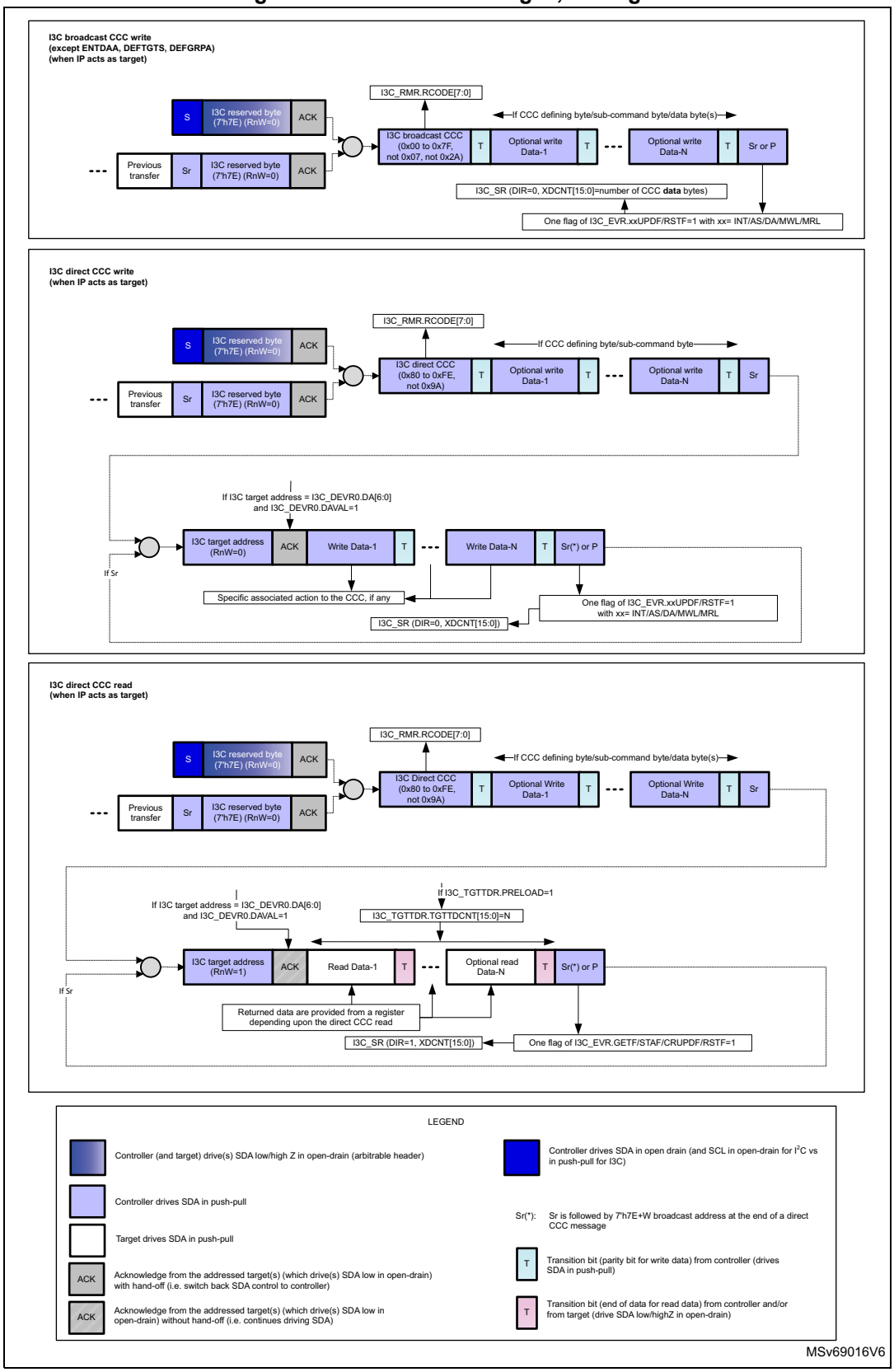
Figure 903. I3C broadcast, direct read and direct write RSTACT CCC, as controller



45.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target

Figure 904 illustrates I3C broadcast CCC write transfer (except ENTDAA, DEFTGTS, DEFGRPA), direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.

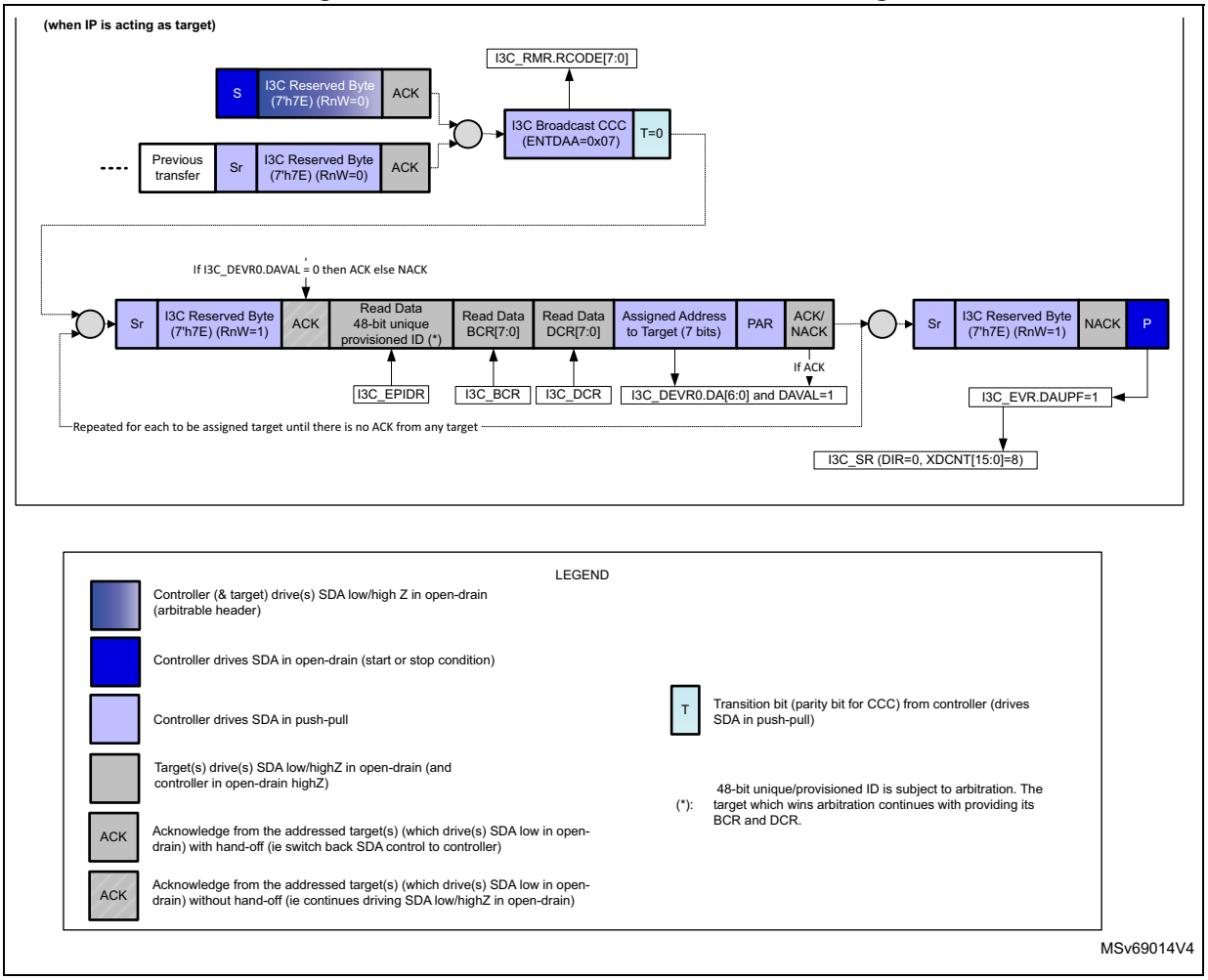
Figure 904. I3C CCC messages, as target



45.9.6 I3C broadcast ENTDAA CCC transfer, as target

Figure 905 illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as target.

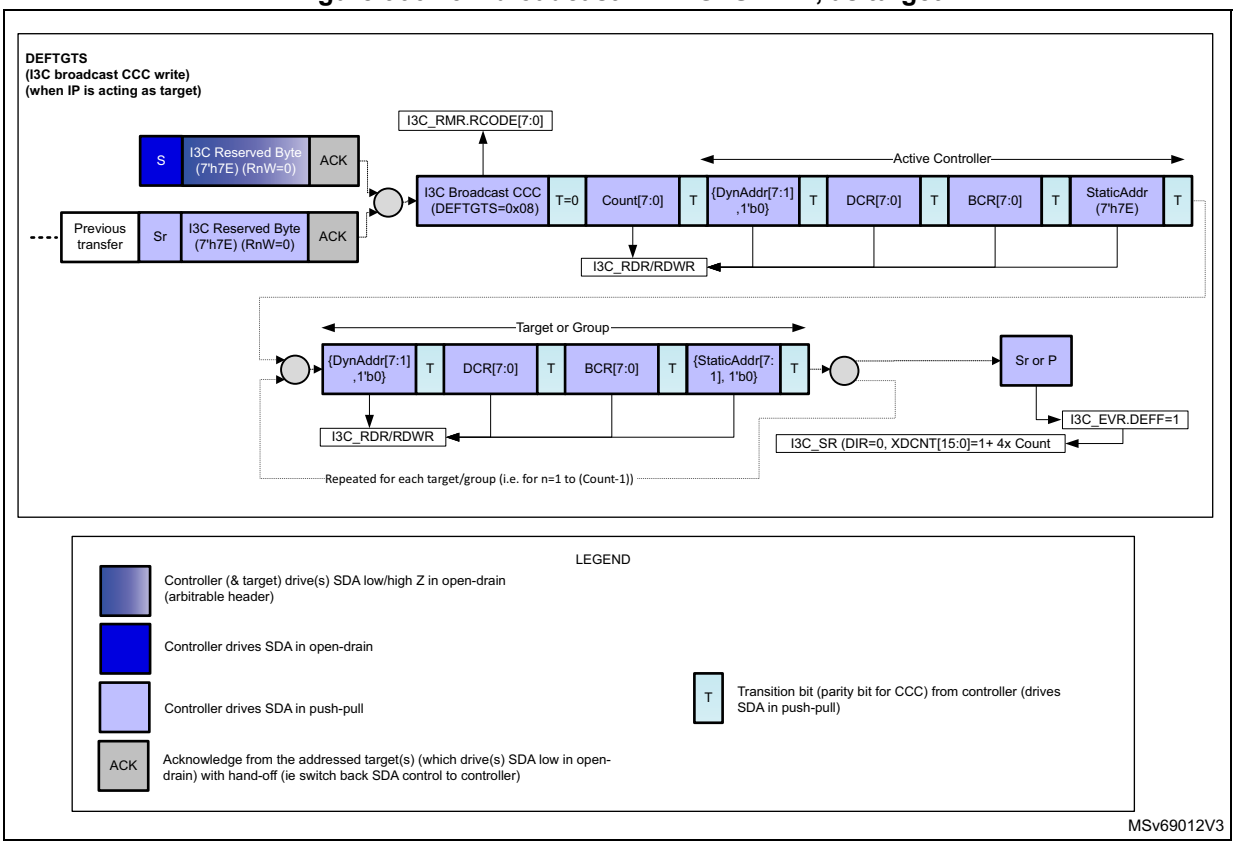
Figure 905. I3C broadcast ENTDAA CCC, as target



45.9.7 I3C broadcast DEFTGTS CCC transfer, as target

Figure 906 illustrates I3C broadcast DEFTGTS CCC, as communicated on the I3C bus, and as programmed when acting as target.

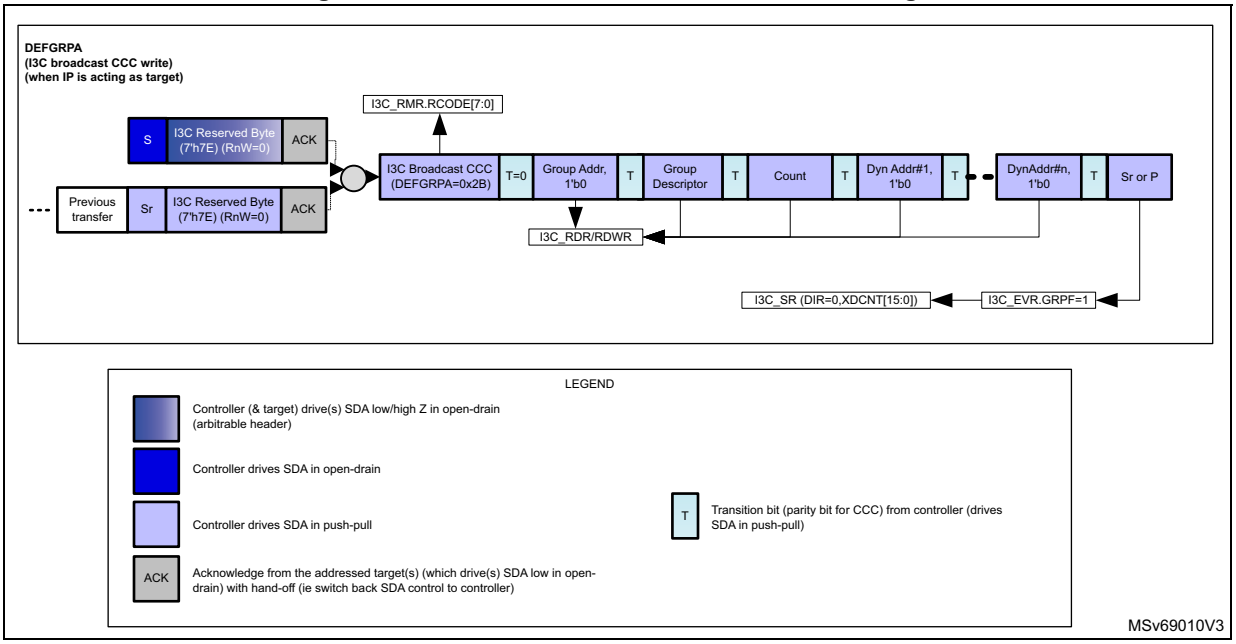
Figure 906. I3C broadcast DEFTGTS CCC, as target



45.9.8 I3C broadcast DEFGRPA CCC transfer, as target

Figure 907 illustrates I3C broadcast DEFGRPA CCC, as communicated on the I3C bus, and as programmed when acting as target.

Figure 907. I3C broadcast DEFGRPA CCC, as target



45.9.9 I3C direct GETSTATUS CCC response, as target

When the I3C acts as target, the hardware returns two data bytes on reception of GETSTATUS CCC, with format 1 (without defining byte or with defining byte TGTSTAT = 0x00), or format 2 (with defining byte PRECR = 0x91).

The returned 2-byte STATUS[15:0] with format 1 on the I3C bus is then as follows:

- STATUS[15:14] = 00 (unused)
- STATUS[13] = 1 if a missed start was detected since the former GETSTATUS CCC, else 0
- STATUS[12] = 1 if an overrun/underrun error was detected since the former GETSTATUS CCC, else 0
- STATUS[11] = 1 if an SCL stable for more than 125 µs was detected during an SDR read since the former GETSTATUS CCC, else 0
- STATUS[10:8] = 000 to 110: encoded value x = 0 to 6, corresponding to a target error TEx if a protocol error was detected since the former GETSTATUS CCC (if STATUS[5] = 1), else 000
- STATUS[7:6] = 00 (ready to prepare for hand-off procedure)
- STATUS[5] = 1 if a protocol error was detected since the former GETSTATUS CCC, else 0
- STATUS[4] = 0 (reserved)
- STATUS[3:1] = 000 (unused)
- STATUS[0] = 1 if there is a pending interrupt (if an IBI is configured in the I3C_CR register, and IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0.register, and the IBI is not yet acknowledged by the controller neither disabled via DISEC), else 0

The returned 2-byte STATUS[15:0] with format 2 on the I3C bus is then as follows:

- STATUS[15:8] = 0000 0000 (unused)
- STATUS[7:2] = 00000 (unused)
- STATUS[1] = 1 if a received DEFTGTS or a received DEFGRPA CCC is still under software processing, and the related event is not yet cleared by software (DEFF = 1 or GRPF = 1 in the I3C_EVR register); the controller must wait before issuing a GETACCR CCC (else it is not acknowledged)
- STATUS[0] = 1 if a DEFTGTS or DEFGRPA CCC may have been missed. This bit is asserted if a missed start is detected (WKPF = 1 in the I3C_EVR register), de-asserted if DEFF = 1 or GRPF = 1 in the I3C_EVR register.

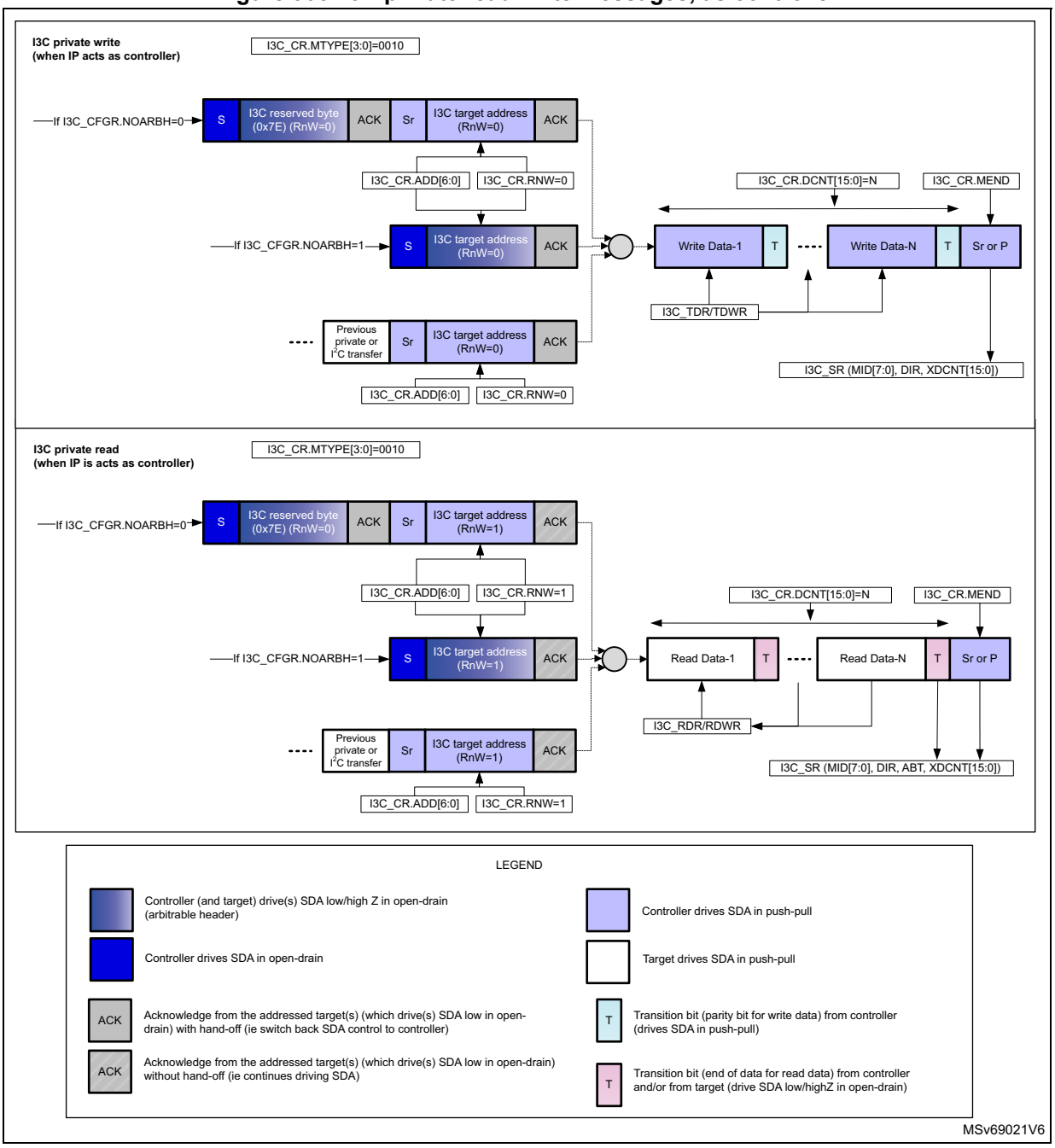
Completion of a GETSTATUS CCC of format 1 is reported by STAF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if STAIE = 1 in the I3C_IER register).

Completion of a GETSTATUS CCC of format 2 is reported by GETF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if GETIE = 1 in the I3C_IER register).

45.9.10 I3C private read/write transfer, as controller

Figure 908 illustrates private read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

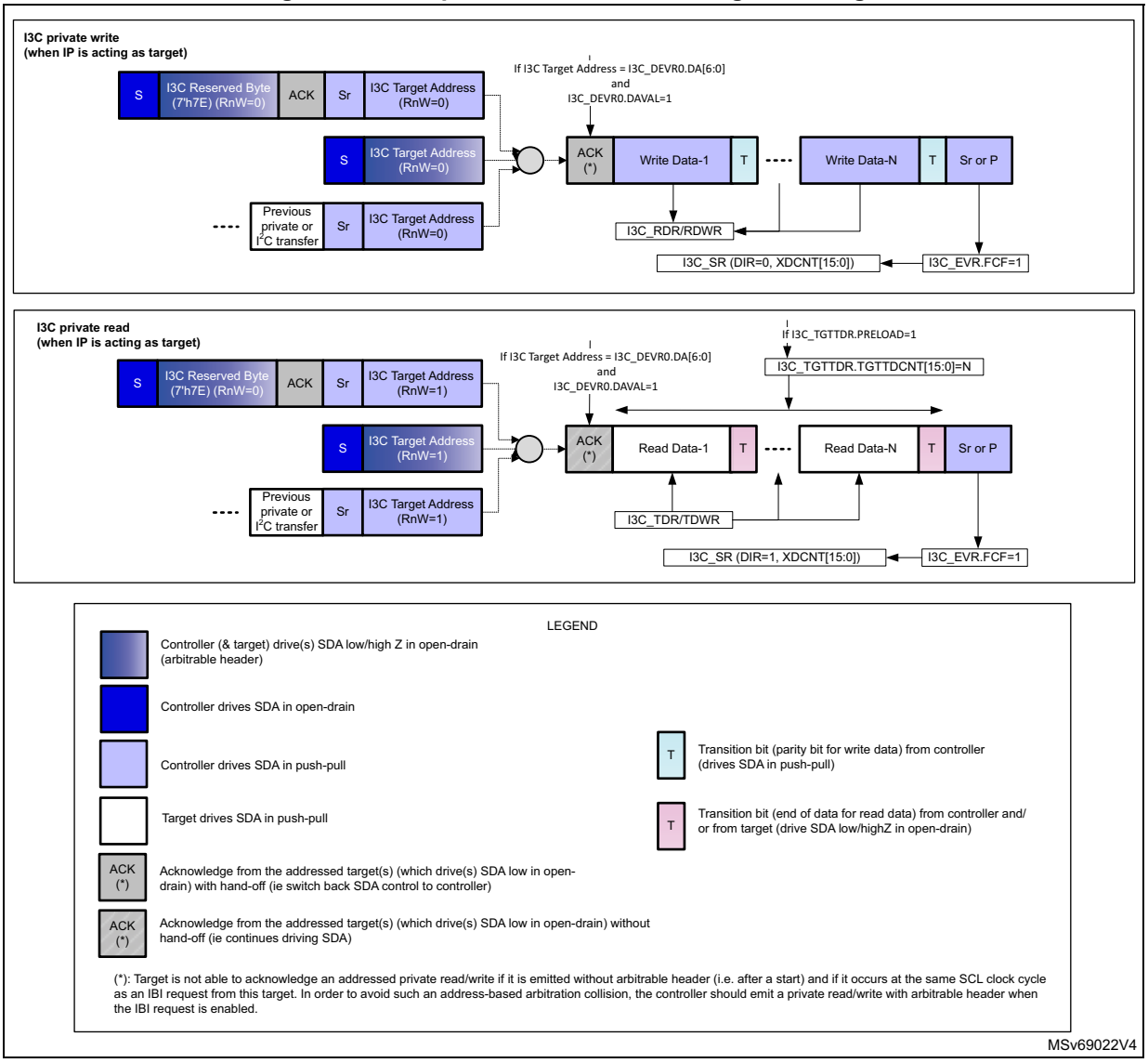
Figure 908. I3C private read/write messages, as controller



45.9.11 I3C private read/write transfer, as target

Figure 909 illustrates I3C private read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.

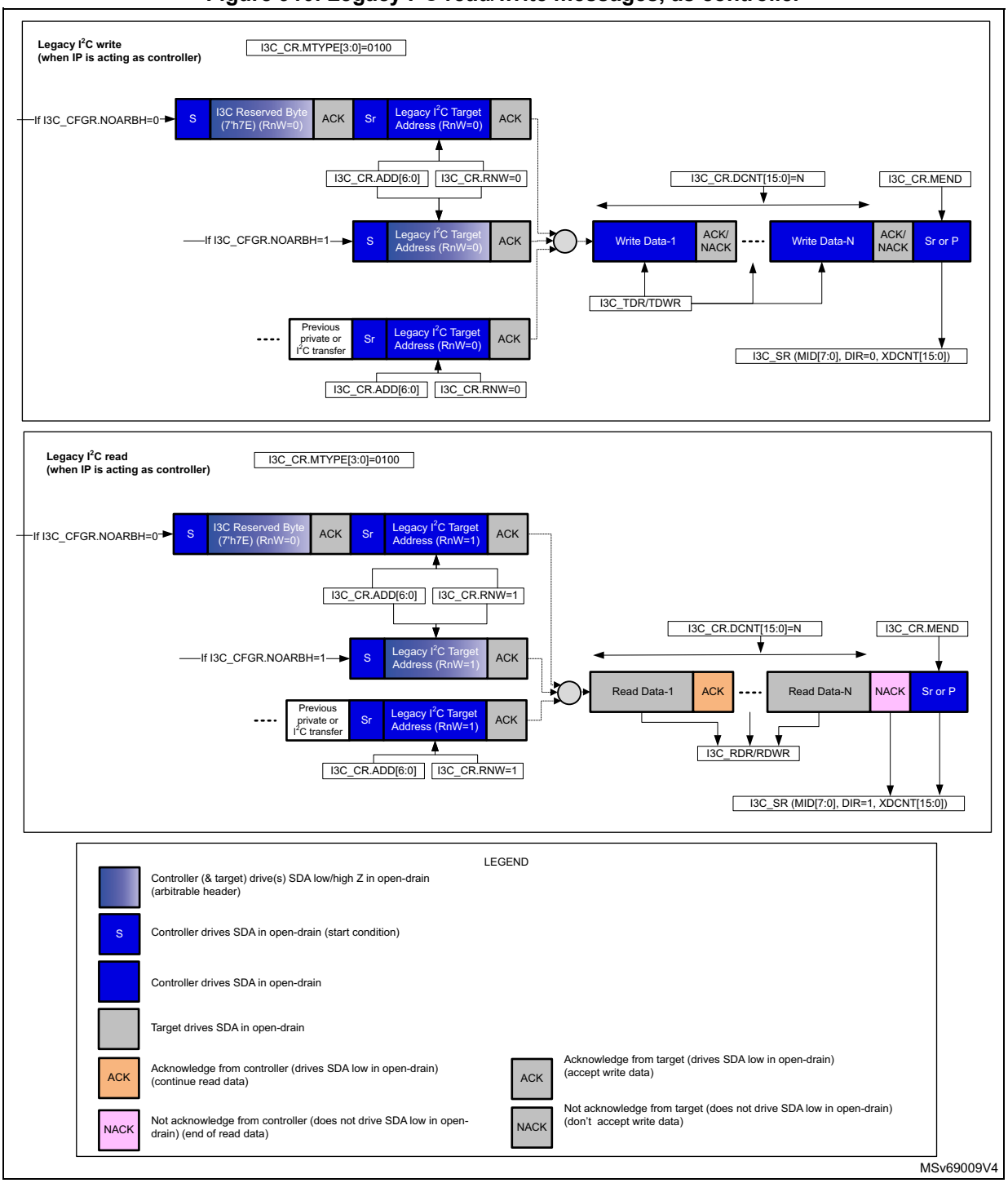
Figure 909. I3C private read/write messages, as target



45.9.12 Legacy I²C read/write transfer, as controller

Figure 910 illustrates legacy I²C read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

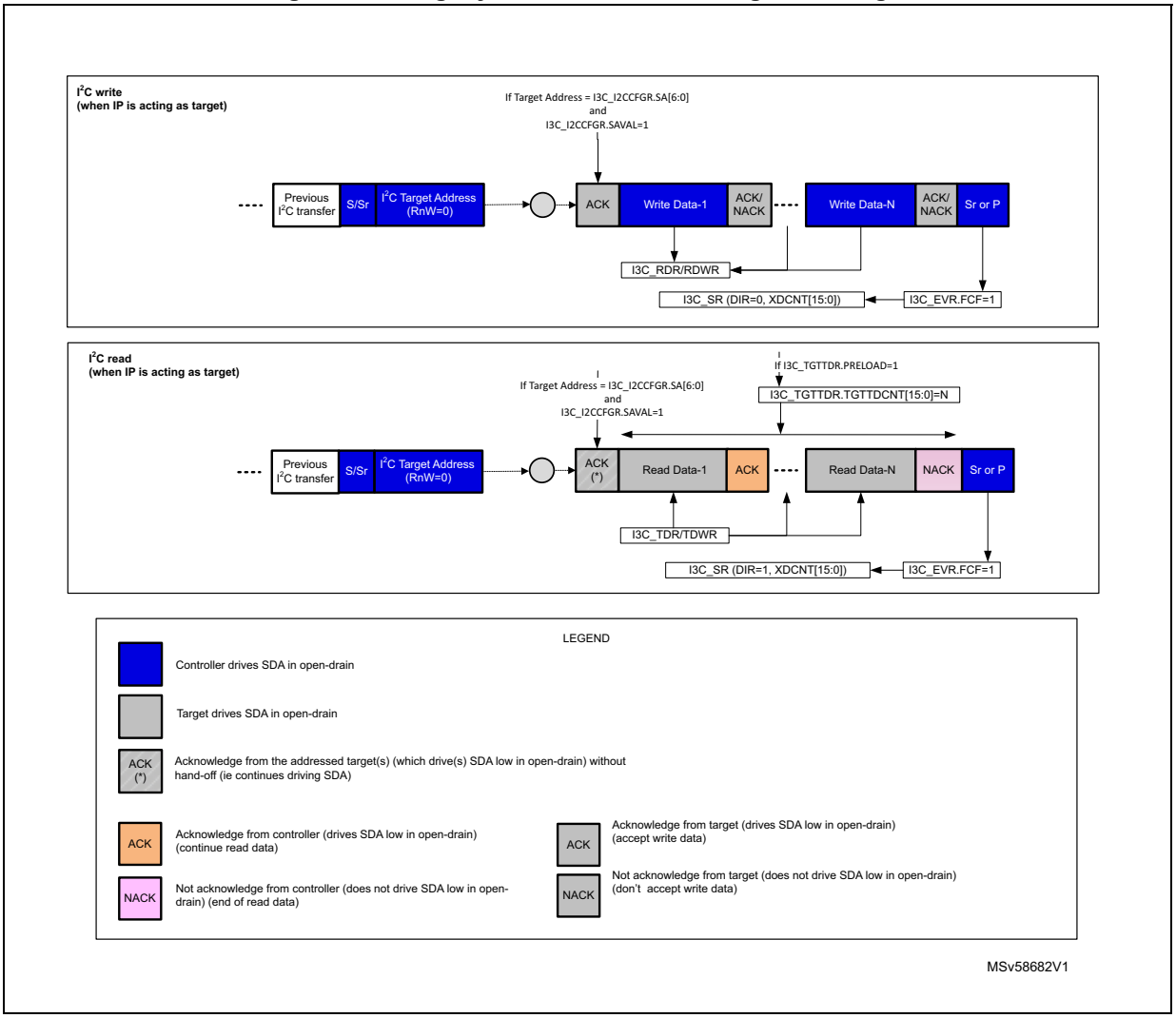
Figure 910. Legacy I²C read/write messages, as controller



45.9.13 Legacy I²C read/write transfer, as target

Figure 911 illustrates legacy I²C read/write transfer, as communicated on the I²C bus, and as programmed when acting as target.

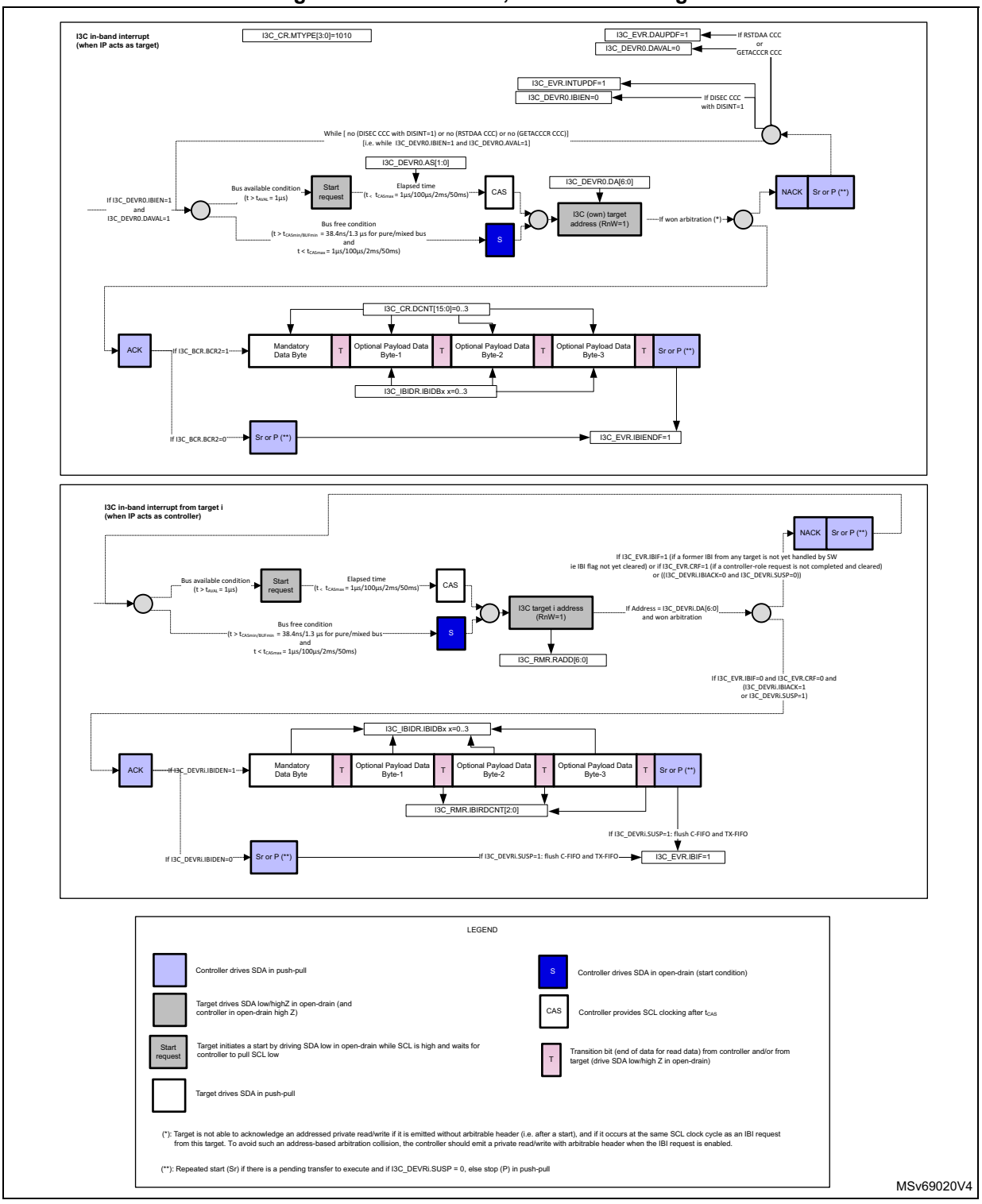
Figure 911. Legacy I²C read/write messages, as target



45.9.14 I3C IBI transfer, as controller/target

Figure 912 illustrates IBI (in-band interrupt) transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 912. IBI transfer, as controller/target



When the I3C peripheral acts as controller, the I3C_IBIDR register is used to receive the IBI data payload. Consequently, the IBI request from the target must not exceed a 4-byte data payload. If there is more information to be exchanged in the context of this in-band interrupt, the controller software must issue a private read.

Timing control and sensor acquisition timestamping when I3C controller

When the I3C peripheral acts a controller, it can use timing control and asynchronous mode 0 to timestamp the moment at which the target(s) acquired sampled data, and to time-correlate them.

To do this, the software must go through the following sequence:

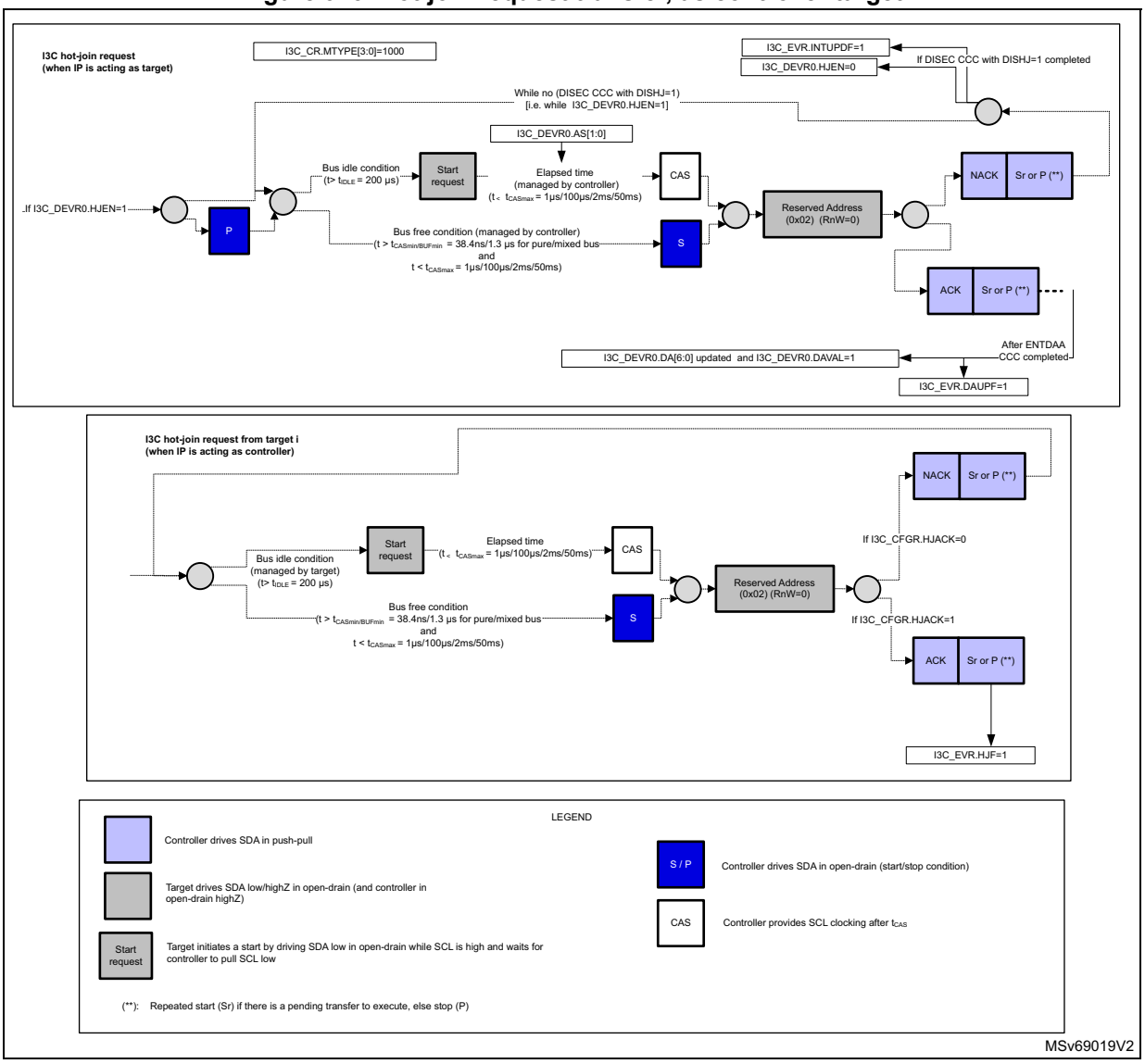
1. Allocate and configure a timer in capture mode, to be connected to the I3C generated event i3c_ibi_ack so that for each IBI acknowledge the counter of the timer can be captured.
2. Program a broadcast/direct SETXTIME CCC with sub-command byte 0xDT to enter asynchronous mode 0 on the target(s) side.
3. After being notified of the command completion by the flag and/or the related interrupt (FCF flag is set in I3C_EVR register), reset and enable the timer to start the counter on the controller side.
4. After being notified that an IBI is complete by the flag and/or the related interrupt (IBIF = 1 in the I3C_EVR register), and if the IBI is with timing info (if IBIDB0[7:5] = 100 in the I3C_IBIDR register), read the captured value of the timer, as C_TIM. Given that:
   a) C_C2 = 9 × (SCLL_PP[7:0] + 1 + SCLH_I3C[7:0] + 1) × T_{I3CCLK}
   b) T_C1 (16-bit value) = {IBIDB2, IBIDB1} in the I3C_IBIDR register
   c) T_C2 (8-bit value) = IBID3 in the I3C_IBIDR register

   Compute the timestamp of the sampled data received from the target, as follows (see MIPI spec formula for more details): C_TS = C_TIM - C_C2 x T_C1 / T_C2.
5. Repeat step *4*, for each received IBI.
6. Generate a broadcast/direct SETXTIME CCC with subcommand byte 0xFF to exit asynchronous mode 0 and disable/deallocate the timer resource.

45.9.15 I3C hot-join request transfer, as controller/target

Figure 913 illustrates hot-join request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

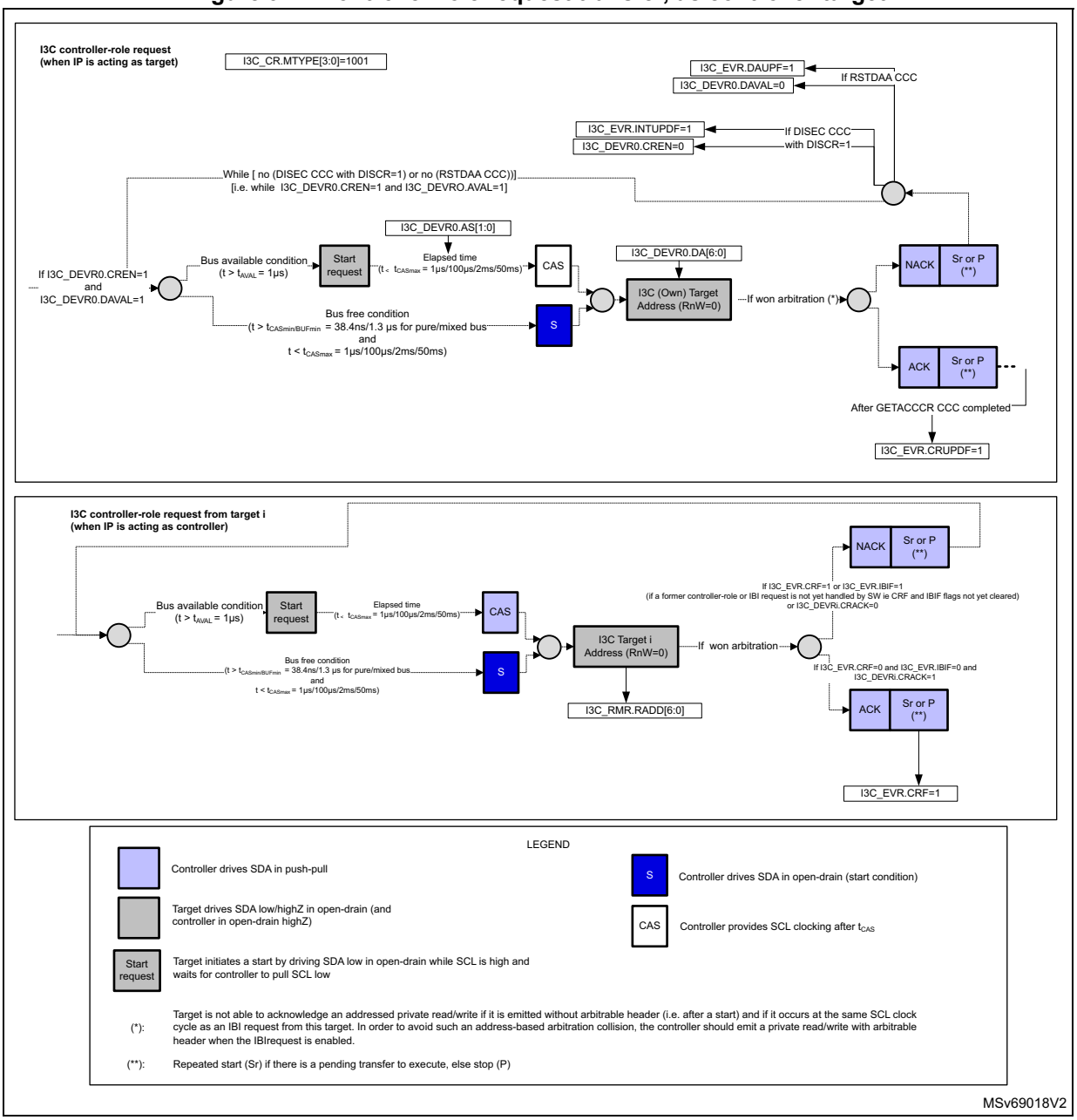
Figure 913. Hot-join request transfer, as controller/target



45.9.16 I3C controller-role request transfer, as controller/target

Figure 914 illustrates controller-role request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 914. Controller-role request transfer, as controller/target



45.10 I3C FIFOs management, as controller

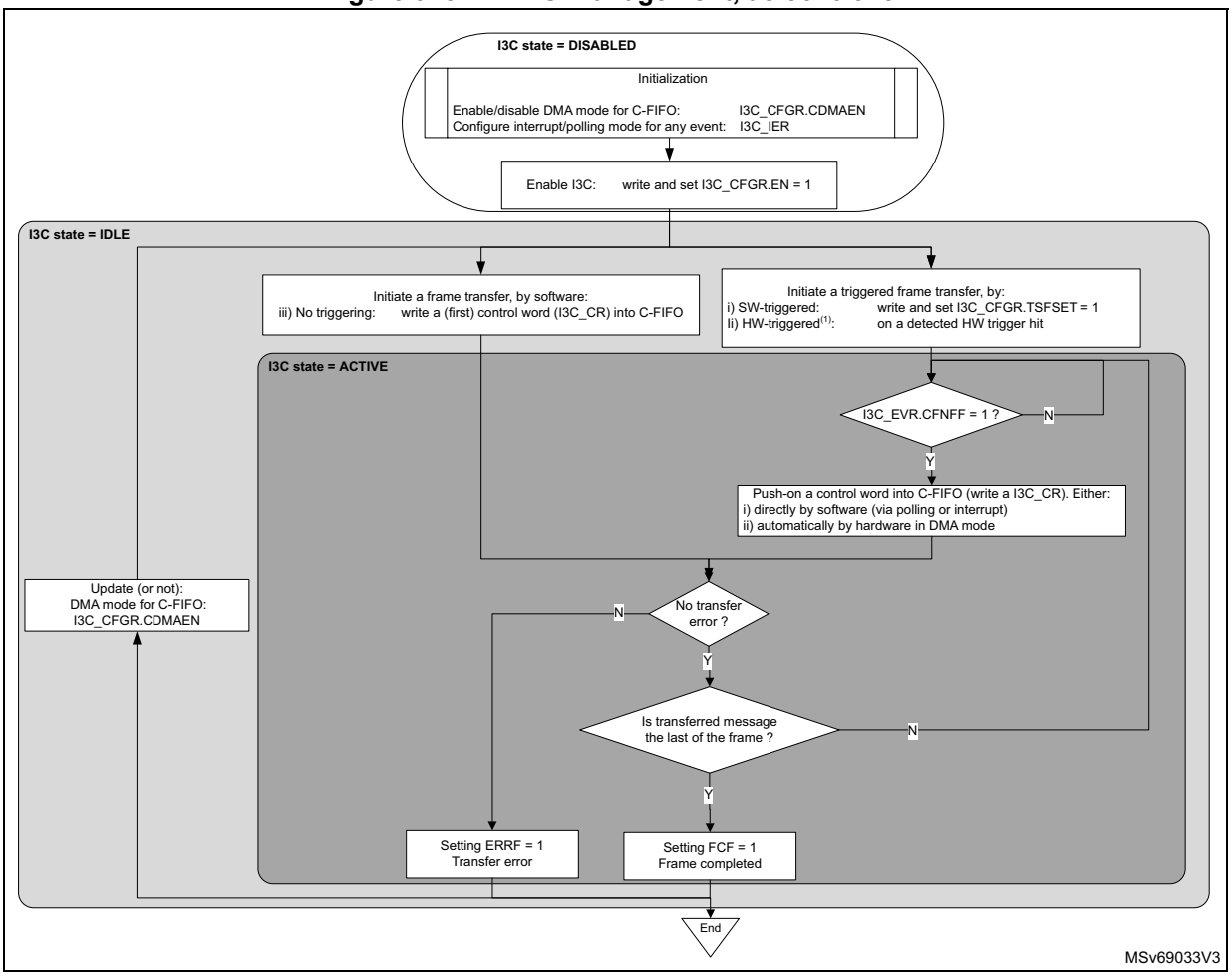
45.10.1 C-FIFO management, as controller

When controller, as illustrated in figures of *Section 45.9*, C-FIFO can be used during any of the following transfers:

- broadcast CCC (*Figure 901*, *Figure 902*, *Figure 903*)
- direct read/write CCC (*Figure 903*)
  - command part, first message
  - data part, next message(s)
- private read/write (*Figure 908*)
- legacy I2C read/write (*Figure 910*)
- software-initiated error recovery (SCL forced to be stopped until next header message followed by HDR exit pattern)

Figure 915 illustrates the management of the C-FIFO for queuing control word(s) on the I3C bus, when the I3C peripheral acts as controller.

Figure 915. C-FIFO management, as controller



1. This feature is implementation-dependent and can be unavailable. Refer to *Section 45.3.4*.

Note: The software can disable the generation of FCF flag by writing 1 into the FCFDIS bit of the I3C_CFGR register. This can be used with hardware triggers and autonomous DMA (if this feature is available, see Section 45.3) to stream a series of frames.

First, the software must initialize the C-FIFO management via CDMAEN in the I3C_CFGR register, to be written either:

- directly by the software (if CDMAEN = 0) at the control word level:
  - via polling mode (CFNFIE = 0 in the I3C_IER register): waiting for a next control word is requested by the hardware (CFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_CR register
  - via enabled interrupt notification if CFNFIE = 1
- by the allocated DMA channel (if CDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tc_dma):
  - as configured at block level, the DMA is automatically pushing-on/writing control word(s) into the I3C_CR register from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

In any case, if C-FIFO is empty and a restart must be emitted with a new control word, a C-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The DMA mode for the C-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1 in the I3C_EVR register), the C-FIFO is flushed automatically by the hardware.

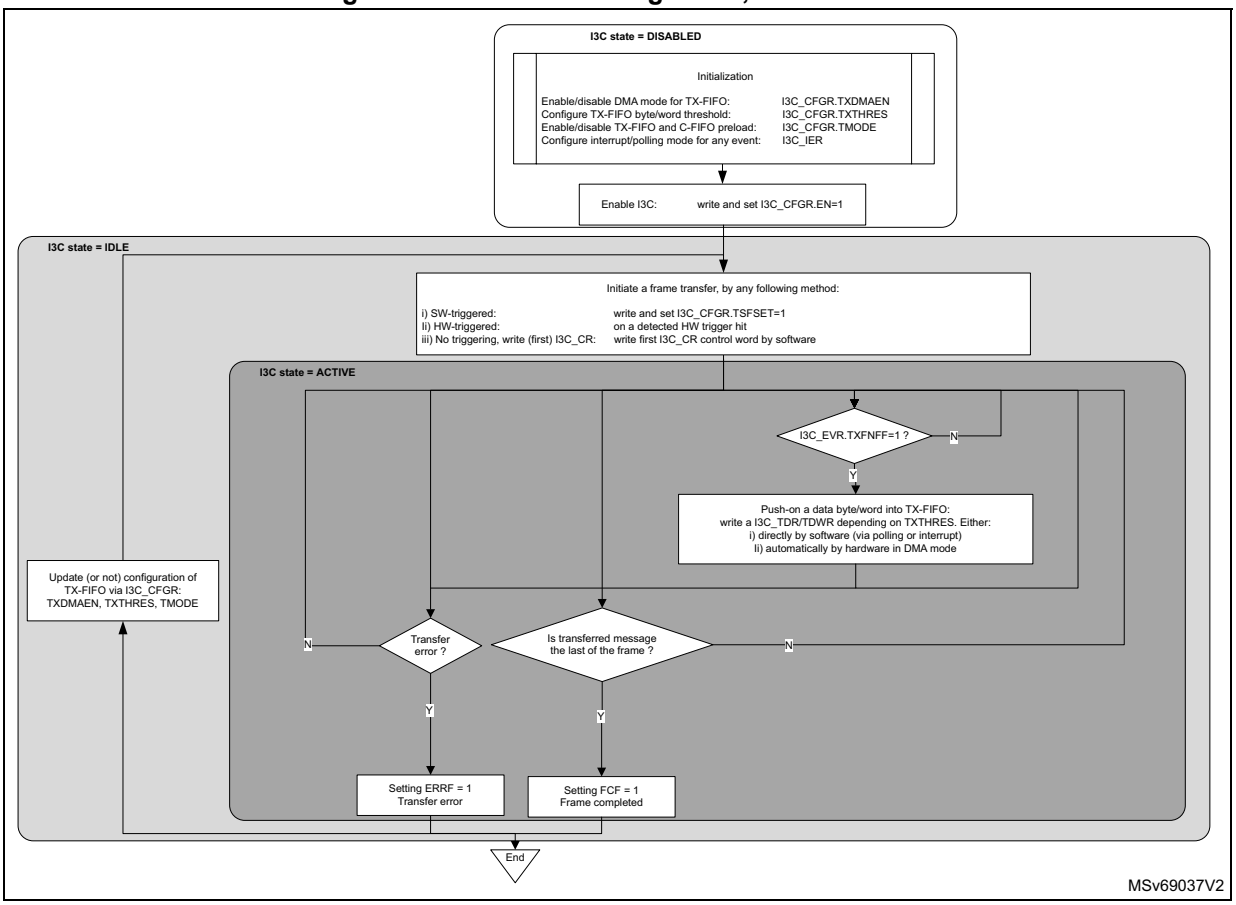
45.10.2 TX-FIFO management, as controller

When controller, as shown in figures of *Section 45.9*, TX-FIFO can be used during any of the following transfers:

- broadcast or direct CCC (including ENTDAA and RSTACT), if a defining/sub-command byte or if data byte(s) are present (*Figure 901*, *Figure 902*, *Figure 903*)
- private write (*Figure 908*)
- legacy I2C write (*Figure 910*)

Figure 916 illustrates the management of the TX-FIFO for queuing data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as controller.

Figure 916. TX-FIFO management, as controller



First, the software must initialize the TX-FIFO management via the following fields in the I3C_CFGR register:

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) into TX-FIFO
- TMODE: enable/disable TX-FIFO and C-FIFO preload

Then, depending upon bit TXDMAEN in the I3C_CFGR register, the TX-FIFO is written either:

- directly by the software (if TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon bit TXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (if TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon bit TXTHRES in the I3C_CGFR. register) from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last data byte/word to be transmitted is flagged by TXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software to update the pointer to the buffer where the byte(s)/word(s) of the next message is/are stored.

When frame completion is reported (FCF = 1 in the I3C_EVR register, and the corresponding interrupt is enabled), the TX-FIFO is empty.

If the TX-FIFO is empty and a data byte must be transmitted on the I3C bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1), the TX-FIFO is automatically flushed by the hardware.

No C-FIFO/TX-FIFO preload

As defined in *Table 520*, C-FIFO size is two words, TX-FIFO size is 8 bytes.

When no C-FIFO/TX-FIFO preload is configured (TMODE = 0 in the I3C_CFGR register), the I3C peripheral emits a start on the I3C bus as soon as the first control word is written into the C-FIFO. Then, it decodes the I3C_CR register, and requires a next data byte/word to be written, if needed within this message. As soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus or if the C-FIFO gets available room), this control word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as another data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus, and if TX-FIFO is not full, and if data byte(s)/word(s) must be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

C-FIFO and TX-FIFO preload

When C-FIFO/TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), before emitting a start on the bus, the I3C peripheral waits for loading as much as possible both the C-FIFO and the TX-FIFO, as follows:

1. Wait for a first control word to be written into the C-FIFO.
2. Wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the first control word (if RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.
3. If TX-FIFO is not full and if the first control word is not the last of the frame (MEND = 0 in the I3C_CR register):
   a) Wait for a second control word to be written into the C-FIFO, then C-FIFO is full.
   b) If TX-FIFO is not full, wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the second control word (RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.

Then, as soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus), this control

word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as a next data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus and if TX-FIFO is not full and if data byte(s)/word(s) are to be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

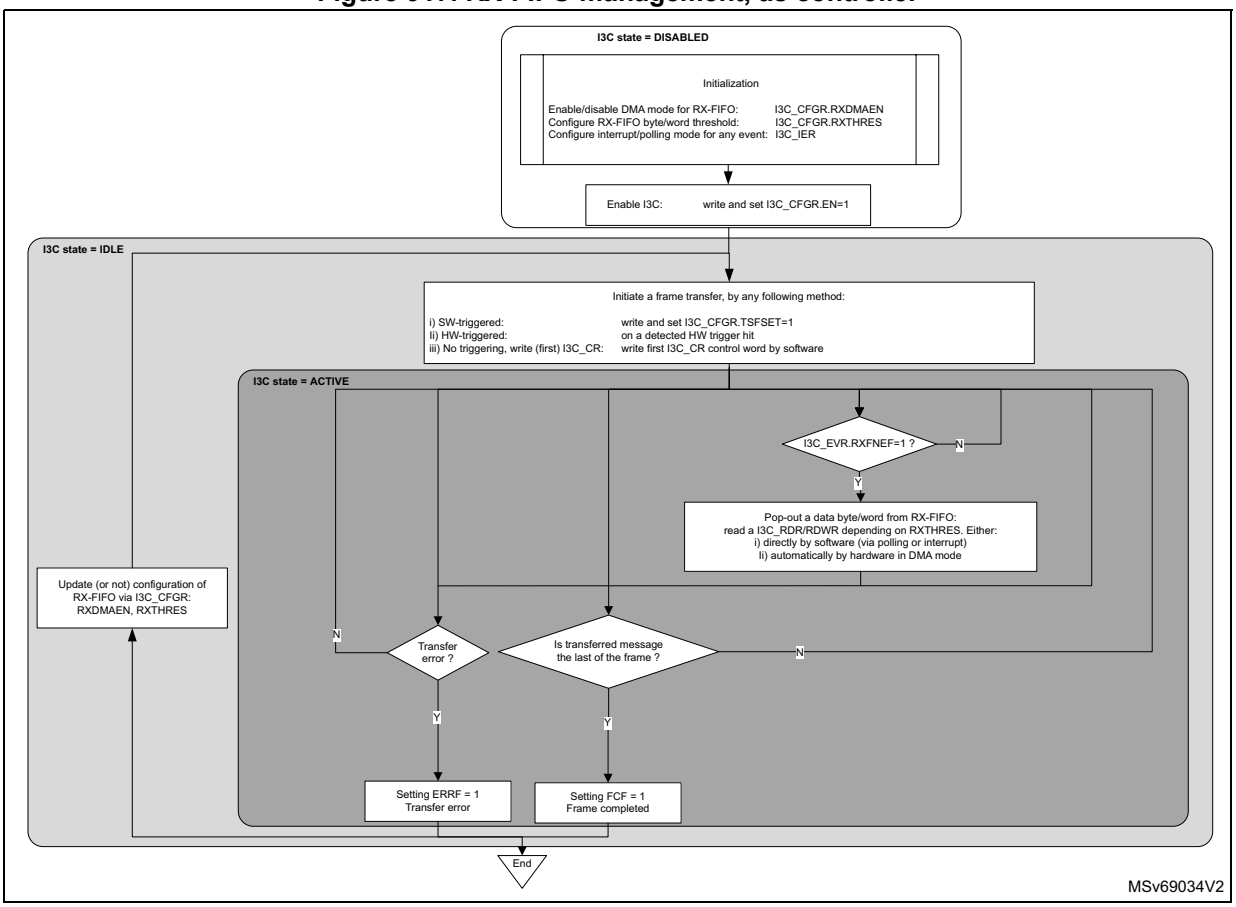
45.10.3 RX-FIFO management, as controller

When controller, as shown in figures of *Section 45.9*, RX-FIFO is used during any of the following transfers:

- broadcast ENTDAA CCC (*Figure 902*)
- direct CCC read (*Figure 901*), including direct RSTACT CCC read (*Figure 903*)
- private read (*Figure 908*)
- legacy I2C read (*Figure 910*)

Figure 917 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received from the I3C bus, when the I3C peripheral acts as controller.

Figure 917. RX-FIFO management, as controller



First, the software must initialize the RX-FIFO management via the following fields of the I3C_CFGR register:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR register, depending upon bit RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1 in the I3C_IER register
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA automatically pops-out/reads data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon bit RXTHRES), and writes them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last received data byte/word from the I3C bus is flagged by IRXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software for updating the pointer to the buffer where is/are stored the data byte(s)/word(s) of the next message.

If RX-FIFO is full and a data byte is received on the I3C bus, an RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

Early read termination from the target

A private read message can be early completed (also known as prematurely ended) by the addressed target.

- If RXDMAEN = 1:
  - the software must allocate, for the DMA request i3c_rx_dma, a DMA channel x, capable of peripheral early termination (refer to DMA implementation section).
  - The software must configure the DMA channel x to enable the DMA peripheral-flow control mode via PFREQ = 1 in the DMA_CxTR2 register, to be able to perform a(n) (early or not) DMA block completion.
  - Then, on block completion, the software can read BR1.BNDT[15:0] in DMA_Cx and/or DMA_CxSAR register(s), to get the effective number of DMA transferred bytes.
- If RXDMAEN = 0:
  - at message level, the last received data byte/word from the I3C bus is flagged by RXFNEF = 1 and RXLASTF = 1 in the I3C_EVR register.

In any case, if the S-FIFO is disabled (if SMODE = 0 in the I3C_CFGR register), the software is notified that an early read termination occurs by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled. Then, software can read the status register I3C_SR to check information related to the last message ,and get the number of received data bytes on the prematurely ended read transfer (XDCNT[15:0] in the I3C_SR register).

In any case, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the software or the DMA (depending upon SDMAEN in the I3C_CFGR register) must read for each message the status register I3C_SR. The number of effective received data bytes on the prematurely ended read message is reported by XDCNT[15:0] (and then ABT = 1) in the I3C_SR register.

For more information, refer to *I3C status register (I3C_SR)* and *Section 45.10.4*.

45.10.4 S-FIFO management, as controller

When controller, S-FIFO can be used by the software to be able to read the *I3C status register (I3C_SR)* for each transferred message.

Reading status register with disabled S-FIFO

If SMODE = 0 in the I3C_CFGR register, the S-FIFO is disabled and the status register can be read as a usual register:

- the register content is overwritten by hardware when is transferee a new message
- I3C_SR contains the status of the last transferred message
- SCL clock is not stalled if status register is not read.

If SMODE = 0, for the specific case of a private read prematurely ended by the target:

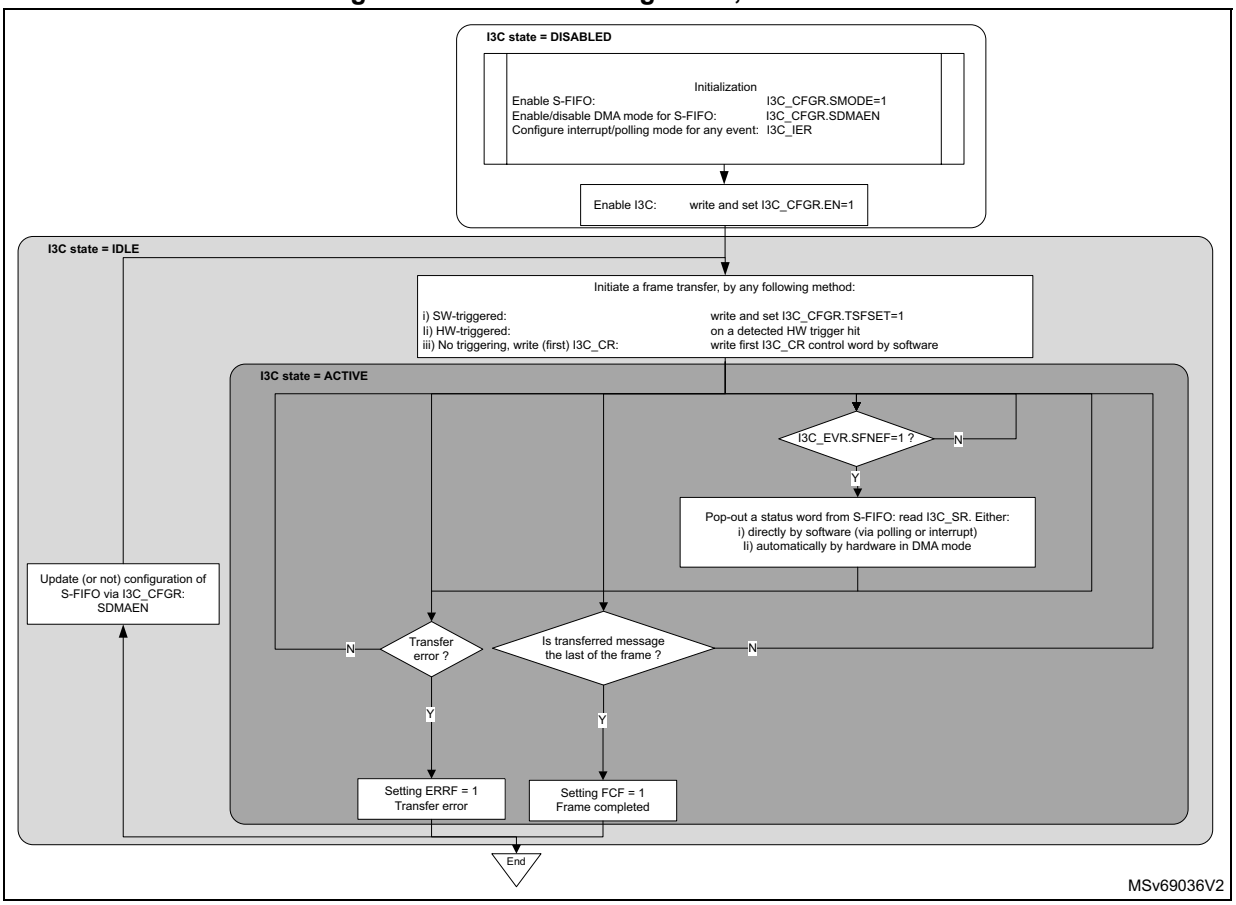
- the software is notified by the flag RXTGTENDF = 1 in the I3C_EVR register and the corresponding interrupt if enabled.
- Then, until that the software has cleared the event flag write and set CRXTGTENDF = 1 in the I3C_CEVR register:
  - no more data byte can be received on the I3C bus and written by the hardware into I3C_RDR/I3C_RDWR registers
  - I3C_SR cannot be updated
  - SCL clock is stalled if needed

Typically, I3C_SR can be read when FCF = 1, or ERRF = 1, or RXTGTENDF = 1 in the I3C_EVR register. XDCNT[15:0] in the I3C_SR register can be read to get the effective number of received data bytes on a private read, after an early termination from the target (refer to *Section 45.10.3*).

Reading status register with enabled S-FIFO

If SMODE = 1 in the I3C_CFGR register, the S-FIFO is enabled. *Figure 918* illustrates the management of the S-FIFO for queuing and popping-out a status word for each executed message on the I3C bus, when the I3C peripheral acts as controller.

Figure 918. S-FIFO management, as controller



First, the software must initialize the S-FIFO management via the SDMAEN field (enable/disable DMA mode for S-FIFO) in the I3C_CFGR register. Then, depending on SDMAEN, the S-FIFO is read either:

- directly by the software (if SDMAEN = 0):
  - via polling mode (SFNEIE = 0 in the I3C_IER register): waiting for a next status word is requested by the hardware (SFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_SR register
  - via enabled interrupt notification (SFNEIE = 1)
- by the allocated DMA channel (if SDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rs_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading status words from the I3C_SR register and writing them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

Each message status must be read, else an overrun error is asserted by the hardware (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register) when the S-FIFO is full and a next message status must be written. The corresponding interrupt is raised if enabled by ERRIE = 1 in the I3C_IER register.

The frame completion (FCF = 1 in the I3C_EVR register) is reported only when the S-FIFO is empty.

The configuration for the S-FIFO management can be modified when the I3C peripheral is not in active state.

45.11 I3C FIFOs management, as target

45.11.1 RX-FIFO management, as target

When target, as shown in figures of *Section 45.9*, the RX-FIFO is used during any of the following received and acknowledged transfers:

- broadcast DEFTGTS CCC (*Figure 906*)
- broadcast DEFGRPA CCC (*Figure 907*)
- private write (*Figure 909*)
- legacy I^2C write (*Figure 911*)

Figure 919 and *Figure 920* illustrate the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received respectively on the I3C or I^2C bus, when the I3C peripheral acts as target.

Figure 919. RX-FIFO management, as target on the I3C bus

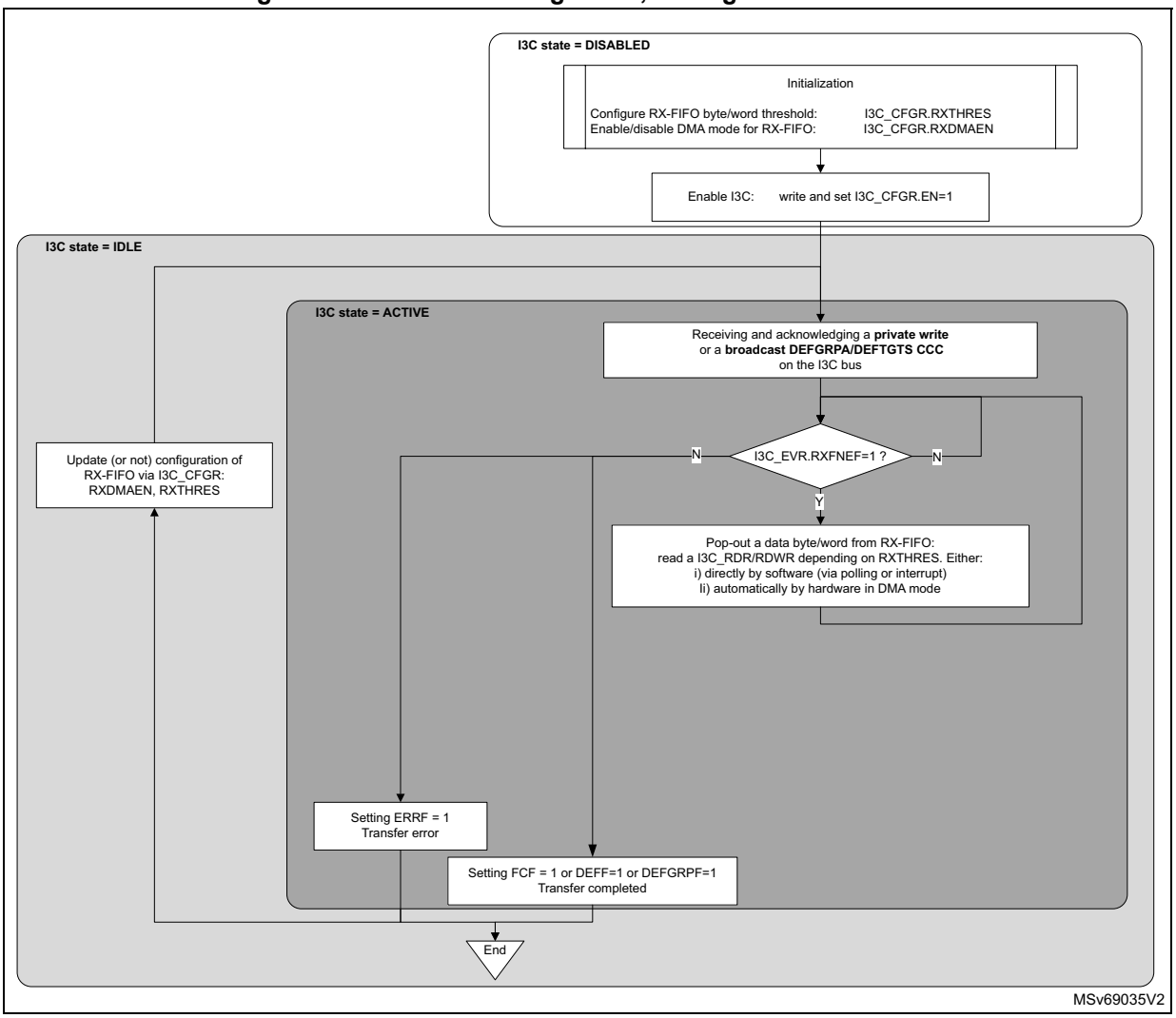
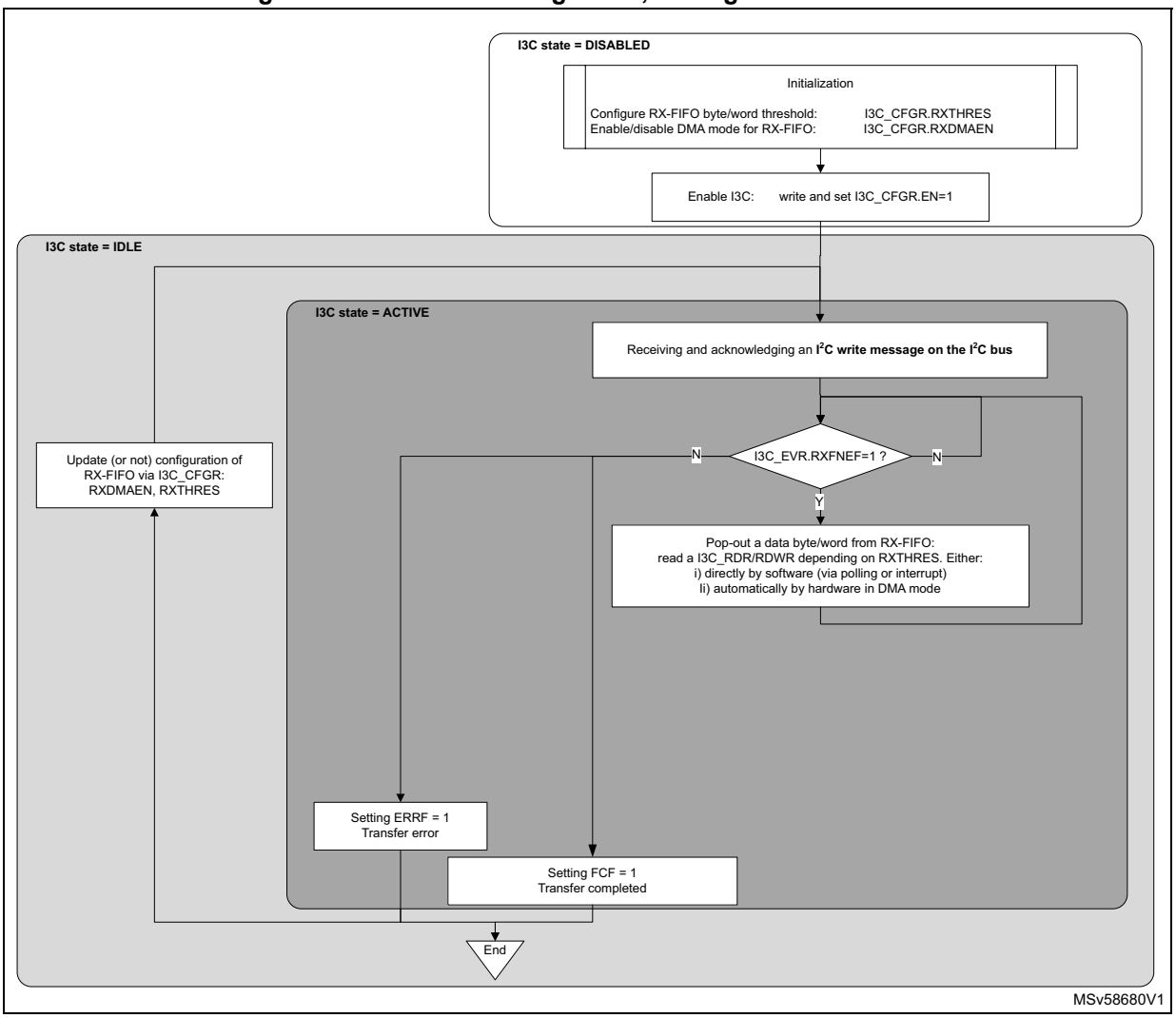


Figure 920. RX-FIFO management, as target on the I²C bus



First, the software must initialize the RX-FIFO management via the following I3C_CFGR register fields:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (if RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR registers, depending upon RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon

RXTHRES) and writing them into its memory destination buffer, until the transfer completion (in the I3C_EVR register, FCF = 1 in case of private write, or GRPF = 1 in case of a DEFGRPA CCC, or DEFF = 1 in case of DEFTGTS CCC), unless a transfer error occurs.

If RX-FIFO is full and a new data byte is received on the I3C bus, a RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

When the transfer is completed (in the I3C_EVR register, FCF = 1, or GRPF = 1, or DEFF = 1):

- RX-FIFO is empty
- Until software has not processed the RX data buffer corresponding to the completed legacy I^2C write or private write / DEFTGTS / DEFGRPA transfer (meaning until software has not cleared the corresponding flag write and set CFCF / CDEFF / CGRPF to 1 in the I3C_CEVR register), if a next private write / DEFTGTS CCC / DEFGRPA CCC is received and a data byte must be written by the hardware into the RX-FIFO, a transfer error is reported with a RX-FIFO overrun (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

45.11.2 TX-FIFO management, as target

Main scheme

When target, as shown in figures of *Section 45.9*, the TX-FIFO is used only during a private read (*Figure 909*).

Figure 921 and *Figure 922* illustrate the management of the TX-FIFO for queuing and pushing-on data bytes or word(s) to be transmitted respectively on the I3C or I^2C bus, when the I3C peripheral acts as target.

Figure 921. TX-FIFO management with I3C_TGTTDR, as target on the I3C bus

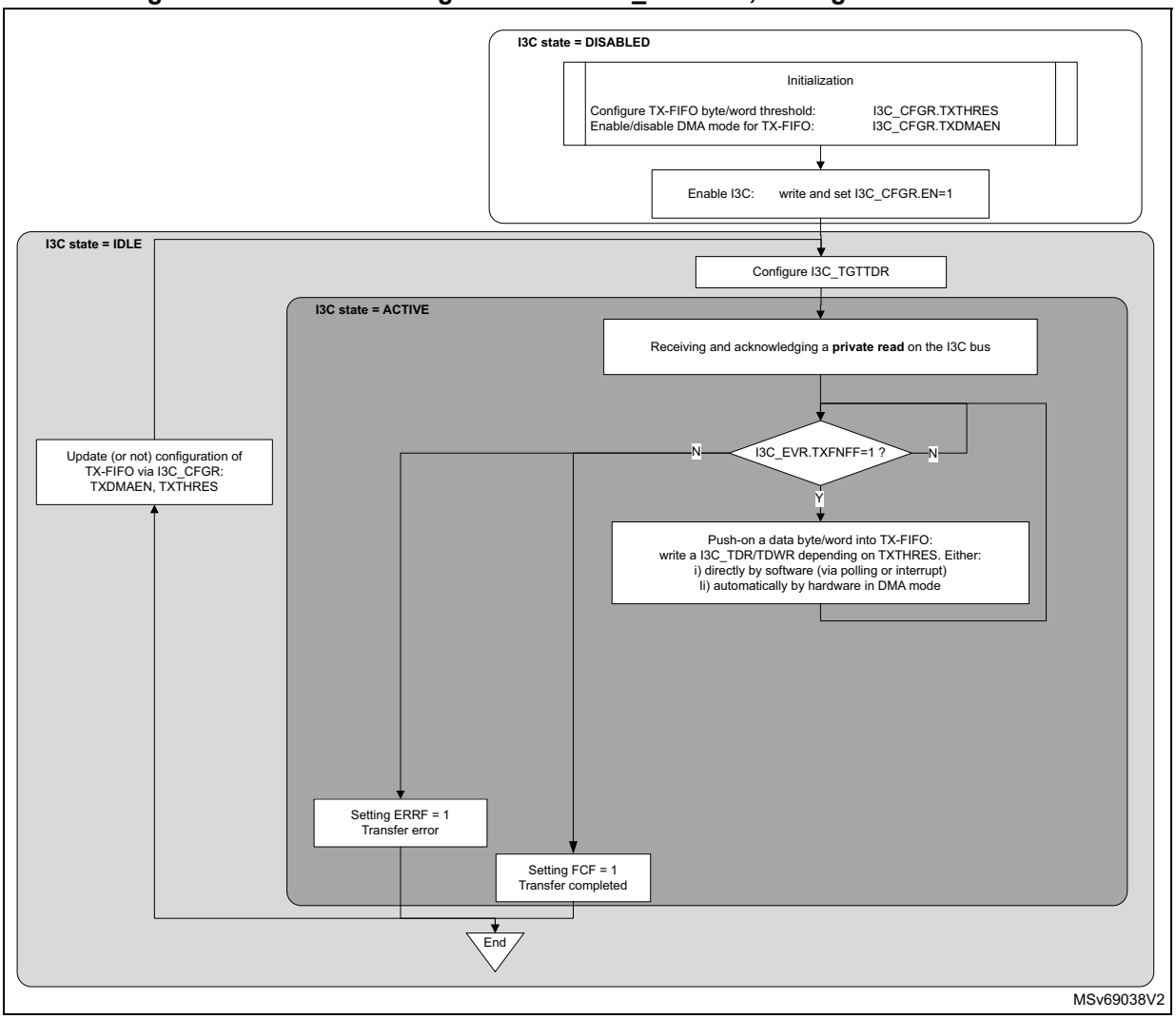
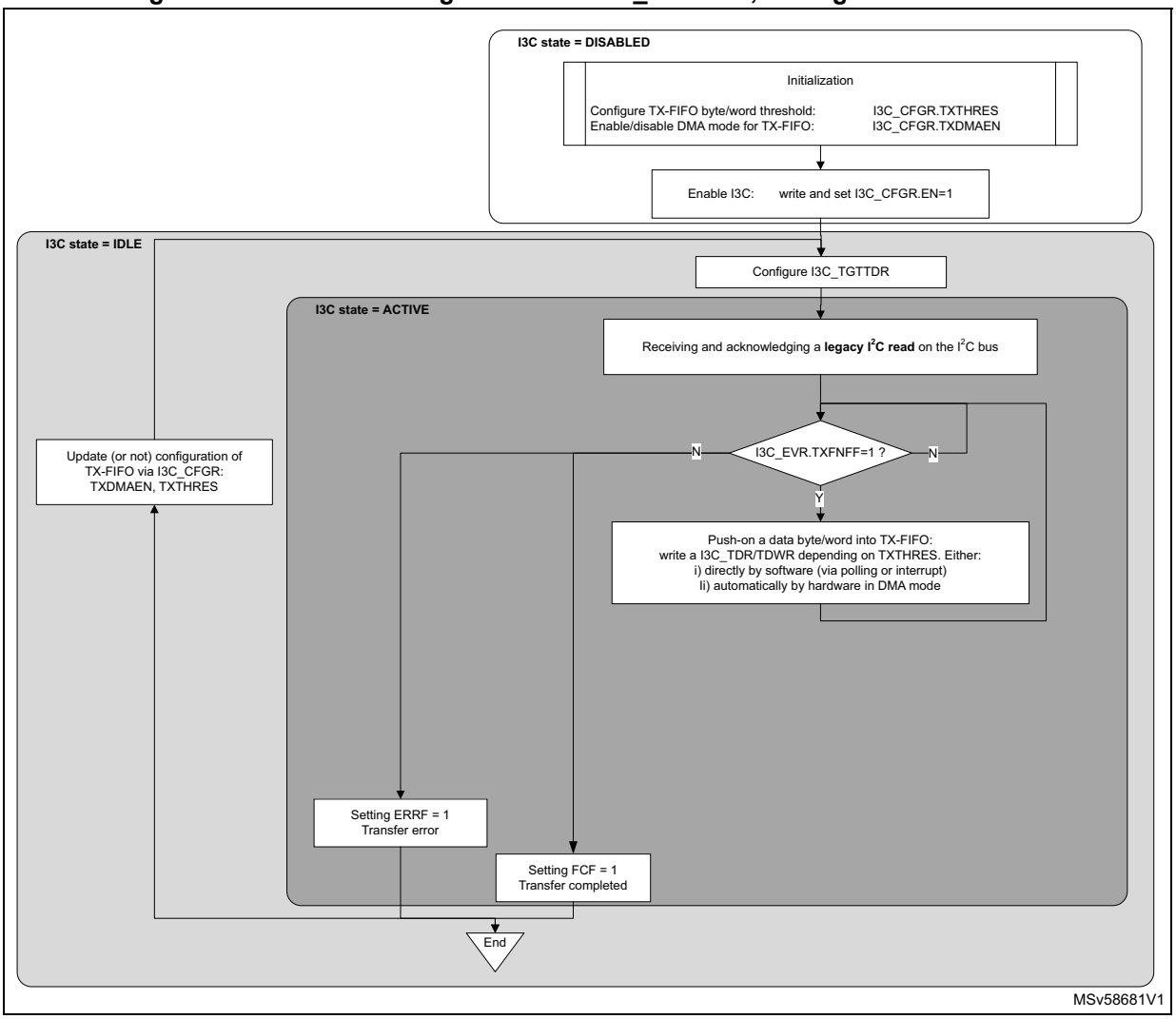


Figure 922. TX-FIFO management with I3C_TGTTDR, as target on the I²C bus



First, the software must initialize the TX-FIFO management and write I3C_CFGR with

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) to TX-FIFO

Then, before receiving a private read on the I3C bus or a legacy I²C read on the I²C bus, the software must configure the *I3C target transmit configuration register (I3C_TGTTDR)* to preload the TX-FIFO with a number of data bytes (write TGTTDCNT[15:0] and PRELOAD = 1 in a single access), so that data bytes from the target are ready to be transmitted on the serial bus:

- If PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size, TX-FIFO is first preloaded up to the FIFO size.
- If PRELOAD = 1 and TGTTDCNT[15:0] ≤ TX-FIFO size, TX-FIFO is preloaded up to TGTTDCNT[15:0].

Note: TX-FIFO size is 8 bytes (refer to Table 520).

Depending upon TXDMAEN, the TX-FIFO is preloaded either:

- directly by the software (TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon TXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon TXTHRES) from its memory source buffer, until the transfer completion (FCF = 1 in the IEC_EVR register), unless a transfer error occurs.

Then, in the same way, either directly by the software or by the allocated DMA channel, if there are remaining TGTTDCNT[15:0] in the I3C_TGTTDR register to be loaded into TX-FIFO for continuing the private read (set PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size), and provided that the private read is not yet completed by the controller:

- If TXTHRES = 0: when a byte is transmitted on the serial bus, a next byte is preloaded into TX-FIFO
- If TXTHRES = 1: when four bytes are transmitted on the serial bus, the next word is preloaded into TX-FIFO.

The private read transfer or the legacy I^2C read transfer is completed (FCF = 1 in the I3C_EVR register) when either the target or the controller first terminates the data byte transfer.

On transfer completion, the software can:

- read XDCNT[15:0] in the I3C_SR register: the effective number of transmitted data bytes
- read TGTTDCNT[15:0] in the I3C_TGTTDR register: the remaining number of bytes to be loaded and transmitted on the serial bus
- flush TX-FIFO: write and set (or not) TXFLUSH = 1 in the I3C_CFGR register, continue (or not) for another read, depending upon the user application

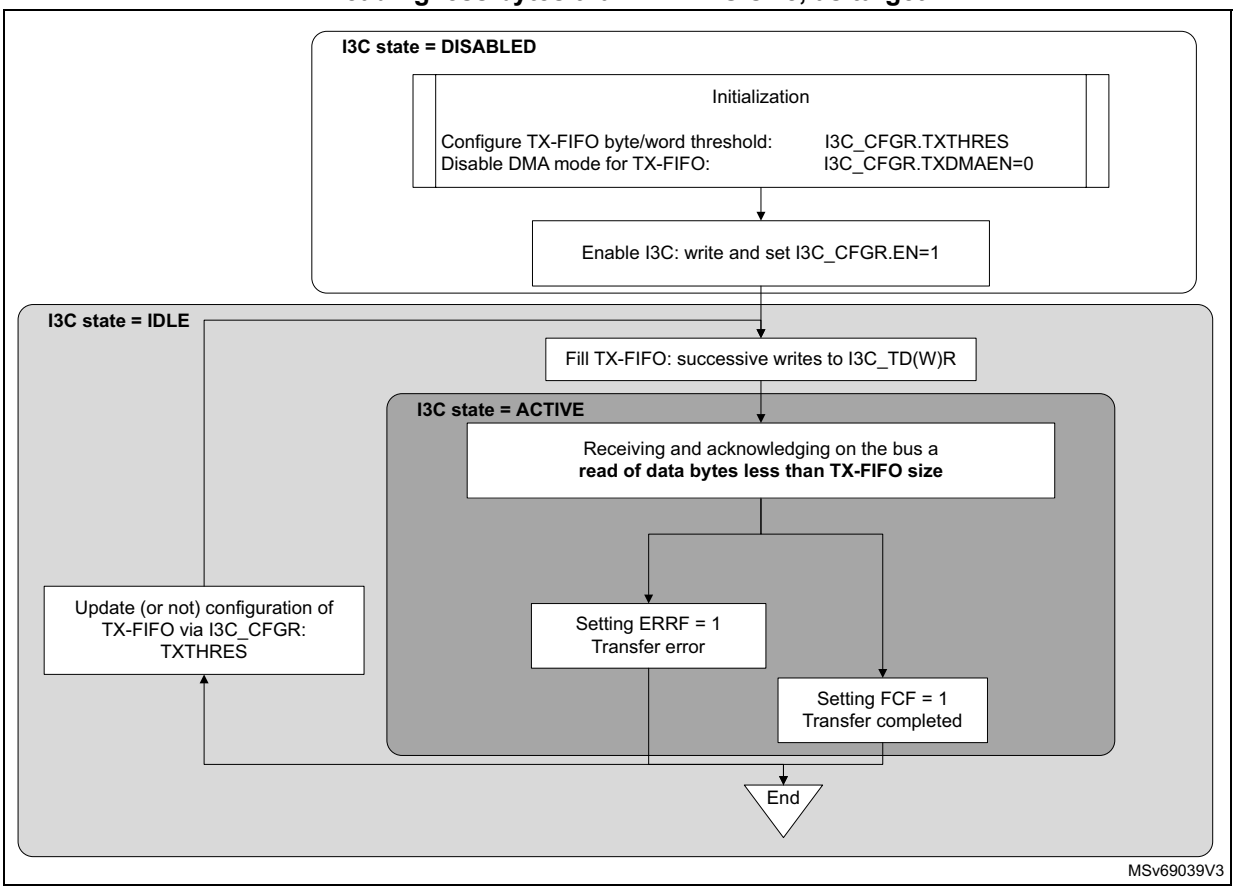
If TX-FIFO is empty and a data byte must be transmitted on the serial bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_EVR register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

Alternative without I3C_TGTTDR, if less bytes than the TX-FIFO size

Alternatively to the use of the I3C_TGTTDR register, as shown in the *Figure 923*, when the DMA is not used (TDMAEN = 0), provided that the number of data bytes to be read on the serial bus is less that the TX-FIFO size, the software can directly prepare and fill the TX-FIFO with the number of bytes to be read directly by the software, by successive writes to the I3C_TDR or I3C_I3C_TDWR, depending upon TXTHRES.

Figure 923. TX-FIFO management by software without I3C_TGTTDR if reading less bytes than TX-FIFO size, as target



45.12 I3C error management

45.12.1 Controller error management

Table 528 enumerates the I3C bus error conditions, and, for each of them, the corresponding detection, action and reporting when the I3C peripheral acts as controller.

Table 528. I3C controller error management

Error type	Description	Error detection	Controller action	Reported error(1)
CE0	Illegally formatted CCC (e.g. less returned data byte(s))	Prematurely ended read data by target on a direct CCC read(2)	Hardware emits a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0000

Table 528. I3C controller error management (continued)

Error type	Description	Error detection	Controller action	Reported error(1)
CE1	Monitoring error	An incorrect ACK is detected at the end of a legacy I^2C read	Hardware keeps SCL running for nine clock cycles and another byte to be possibly exchanged, then emits again a NACK, followed by a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0001
		Restart or stop cannot be generated at the end of an I3C SDR read	SCL is kept running. Software can stop SCL by writing a control word with MTYPE[3:0] = 0000 in the I3C_CR register (header message), then must typically wait at least 150 µs before emitting another message.	
		After a CE1 error, a start cannot be generated		
CE2	No response to broadcast address (0b111_1110)	Header (0b111_1110 + RnW = 0) is detected as NACK-ed during a message different from escalation fault or reset pattern message	Hardware emits an HDR exit pattern followed by a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0010
CE3	Failed controller hand-off	New controller has not driven SCL low after having SDA low (after a start via a test header or a start request from a target) and after the delay time defined by its activity state has elapsed	Hardware emits a start + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0011
-	Incorrect returned 7-bit target address with parity bit on a GETACCCR CCC	Incorrect dynamic address and/or parity bit is detected	Hardware cancels the GETACCCR CCC by emitting a Restart + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.	ERRF = 1 and DERR = 1
-	Addressed target is NACK-ed on a direct CCC read for the first time	NACK is detected	Hardware performs a single-retry by emitting a Restart + 7-bit same target address + RnW = 1.	-
-	Addressed target is NACK-ed on either a direct CCC write, an I3C private read/write, a legacy I^2C, or a direct CCC read for the second time	NACK is detected	Hardware emits a stop.	ERRF = 1 and ANACK= 1

Table 528. I3C controller error management (continued)

Error type	Description	Error detection	Controller action	Reported error(1)
-	Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the first time	NACK is detected	Hardware performs a single-retry assignation loop by emitting a Restart + 0b111_1110 + RnW = 1, followed by an ACK and 8-byte read data from the (most-priority) target, followed by the assigned address + parity bit.	-
-	Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the second time	NACK is detected	Hardware emits a stop.	ERRF = 1 and DNACK= 1
-	Write data is NACK-ed on a legacy I2C write			
-	Control word, status word, transmitted data or read data is not written/read in time vs. I3C bus timings	SCL stall timeout	Hardware emits a stop.	ERRF = 1 and (COVR = 1 or DOVR = 1)

1. ERRF in the I3C_EVR register, PERR, CODERR[3:0], DERR, ANACK, DNACK, COVR, and DOVR in the I3C_SER register.
2. MIPI v1.1: on a GETCAPS CCC, the number of received data bytes can be 2, 3 or 4. However a target compliant with MIPI v1.0 can return only the first byte as per previously named GETHDRCAP CCC. As a result, CE0 is not generated if the number is lower than 4.
On a GETMXDS CCC, the number of received data bytes can be 2 or 5. CE0 is not generated if the number is lower than 5.

45.12.2 Target error management

Table 529 enumerates the I3C bus error conditions, the corresponding detection, action, and reporting when the I3C peripheral acts as target.

Table 529. I3C target error management

Error type	Description	Error detection	Next (when emitted by the controller)	Reported error(1)
TE0	Invalid broadcast address (0b111_1110+RnW = 0) or Invalid 7-bit dynamic address + RnW = 1 after DAA assignment	A forbidden address is detected after a start or a repeated start	Hardware waits for an HDR exit pattern	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1000
TE1	CCC code	A CCC code is detected with a parity error		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1001

Table 529. I3C target error management (continued)

Error type	Description	Error detection	Next (when emitted by the controller)	Reported error(1)
TE2	Write data	A write data byte is detected with a parity error on an I3C private write message	Hardware waits for a stop or a repeated start	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1010
TE3	Assigned address during dynamic address arbitration	Assigned dynamic address is detected with a parity error on a ENTDAA CCC		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1011
TE4	0b111_1110 + RnW = 1 missing after Sr during dynamic address arbitration	Missing {0b111_1110 + RnW = 1} after Sr during dynamic address arbitration is detected		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1100
TE5	Transaction after detecting CCC or invalid I^2C frame	Invalid CCC direction is detected vs. direction provided during address phase, on a CCC direct read/write or misplaced start/stop during a data transfer on an I^2C bus		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1101
TE6	Monitoring error of I3C or I^2C bus	SDA is detected as driven at unexpected value on an data read (during a CCC direct read or a private read or a legacy I^2C read or an IBI)	Hardware releases SDA, then waits for a stop or a repeated start	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1110
-	Monitoring SCL on an SDR data read	SCL is detected as stable for more than 125 µs on an SDR data read (during a CCC direct read or a private read or an IBI)		ERRF = 1 and STALL = 1
-	Data to be transmitted not written/available in time vs. I3C or I^2C bus timings	Data to be transmitted not written by software or DMA in due time on a private read or a legacy I^2C read	Hardware detects a stop or a repeated start	ERRF = 1 and DOVR = 1
-	Received data cannot be registered in time vs. I3C or I^2C bus timings	Received data is not read by software or DMA in due time on a private write or a DEFTGTS or a DEFGRPA CCC or a legacy I^2C write		

1. ERRF in the I3C_EVR register, PERR, ODERR[3:0], STALL, and DOVR in the I3C_SER register.

45.13 I3C wake-up from low-power mode(s)

45.13.1 Wake-up from Stop

The user must first configure the reset and clock controller (RCC) to set up the clock data path down to the I3C peripheral: to select the source oscillator for the I3C kernel clock, the source oscillator for the I3C APB clock, and to set the clocks frequency. At initialization via the RCC, the user must enable the I3C clocks for a given I3C peripheral to be functional, separately for Run/Sleep mode and for Stop mode. For more details about RCC programming, refer to Section Reset and clock control (RCC).

The I3C hardware automatically manages its own clocks gating and generates a separated clock request output signal to the RCC for its kernel clock and its APB clock, whenever the device is in Run, Sleep or Stop mode.

When entering a Stop mode, the V_{CORE} domain is supplied, by default any clock oscillator is disabled, and in any case, neither the system clock nor a peripheral clock is running in the domain.

As controller: wake-up on an IBI without MDB, a hot-join request or a controller-role request

When the peripheral acts as controller, before the product enters a low-power mode, the software must issue an ENTASx CCC, generally with x = 0, 1, 2 or 3, to inform targets that the I3C controller is not expected to exit from idle state neither to communicate on the I3C bus before an interval of, respectively, 1 µs, 100 µs, 2 ms or 50 ms, has elapsed. This delay defines the T_{CAS} delay for the controller to set the SCL bus clock low and running, after a start condition. More specifically for a Stop mode, the CCC must be restricted to an ENTASx with the value x = 1, 2, or 3.

The general scheme for the I3C wake-up from Stop is the following:

1. First the I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the I3C bus (SDA line is detected to be driven low while SCL is high).
   - As controller, as initiated by an external I3C target device for a hot-join/in-band interrupt/controller-role request. Once the kernel clock is provided and running, the I3C hardware uses an internal timer to wait for the corresponding T_{CAS} time to elapse, then drives low SCL and continues toggling SCL to let the target perform its hot-join/in-band interrupt/controller-role request on the I3C bus.
   - On a detected hardware trigger to initiate an I3C transfer on the I3C bus (refer to *Section 45.3.4: I3C triggers* to determine whether the trigger feature is supported).
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds from HSI). The user can keep the source oscillator ON in Stop mode to reduce this startup latency, at the expense of power consumption.
   - The I3C peripheral maintains the kernel clock request until the generation of the Stop condition on the I3C bus.
3. After that the kernel clock is running, as controller the I3C peripheral requests its APB clock to the system:
   - On the ACKed address of a received IBI request (*Figure 912*), and it maintains the APB clock request until that IBIF is cleared.

     If the IBI is without MDB: a Stop is normally generated on the I3C bus, even if the APB clock is not yet provided.
   - On the ACKed address of a received controller-role request (*Figure 913*), and it maintains the APB clock request until that CRF is cleared.
   - On the ACKed address of a hot-join request (*Figure 914*), and it maintains the APB clock request until that HJF is cleared.
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken.
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock that drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as

the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after a delay.

7. With the APB clock running, the I3C peripheral can log the I3C transfer in its status and data registers. When the bus transfer is completed, the I3C peripheral generates the corresponding flag (IBIF/CRF/HJF), and the enabled interrupt can wake up the CPU.

As target: wake-up on a I3C reset pattern

The target reset pattern is a specific scheme for a controller to wake up and possibly reset a target from a low power mode. It consists both in the RSTACT CCC and in the in-band reset pattern generation.

As target, the sequence for the I3C wake-up from Stop on a reset pattern is the following:

1. When the I3C peripheral detects a reset pattern on the bus (14 SDA transitions while SCL is kept low), it requests its APB clock to the system to set the RSTF flag of the I3C_EVR register
2. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
3. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
4. The system enables the source oscillator of the system clock, which gets ready after few microseconds.
5. With the APB clock running, the I3C peripheral can raise the RSTF flag of the I3C_EVR register, and the enabled interrupt can wake up the CPU.

As target: wake-up on a missed start on the serial bus

1. The I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the serial bus (SDA line is detected to be driven low while SCL is high)
   - If the kernel clock is not provided before that SCL is driven low by the external controller, then the I3C target detects that a bus start is missed and waits for the kernel clock to be provided. A controller message that is specifically addressed to this low-power device may have been missed (and NAcked). However the I3C peripheral can also detect a missed start if the controller started a CCC or private or legacy I^2C message to another addressed target, or if another target initiated a start for an IBI/CR/HJ request.
   - On a detected hardware trigger to initiate an I3C IBI/CR/HJ transfer on the I3C bus (refer to *Section 45.3.4: I3C triggers* to determine whether the trigger feature is supported).
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds if HSI). The user may keep the source oscillator ON in

Stop mode to reduce this startup latency, at the expense of power consumption. Especially if the controller is in an Activity State of x = 0 (with a 1 µs T_{CAS} latency).

3. After that the kernel clock is running, as target, the I3C peripheral requests its APB clock to the system to raise the WKPF flag:
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock, which drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after few microseconds.
7. With the APB clock running:
   – The missed start flag (WKPF of the I3C_EVR register) is raised if it occurred, and the enabled interrupt can wake up the CPU. It is then known that a serial bus transaction was missed, but not whether the target was addressed or not. The software uses a timeout to return to Stop mode, if it is not addressed by the controller again within this interval.

45.14 I3C in low-power modes

Table 530. Effect of low-power modes

Mode	Description
Sleep	No effect. I3C interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the I3C registers is kept when entering Stop mode. I3C hardware manages automatically its own clocks gating and generates, for its kernel clock and APB clock, a clock request output signal to the RCC. I3C interrupts can cause the device to wake up and exit Stop mode.
Standby	The I3C peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 45.3.2: I3C wake-up from low-power mode(s)* to see which, if any, Stop modes are supported.

45.15 I3C interrupts

45.16 I3C registers

The I3C registers must be accessed with a 32-bit word aligned address.

Note: I3C_RDR and I3C_TDR registers must be accessed with a single significant LSB data byte for, respectively, reading RX-FIFO and writing TX-FIFO.

45.16.1 I3C message control register (I3C_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This register must be used to control the message to emit on the I3C bus:

- when I3C acts as controller (bit[30] = MTYPE[3] = 0): if there is no CCC code to be emitted bits[29:27] = MTYPE[2:0] differ from 110; else the alternate register description *Section 45.16.2* must be considered.
- when I3C acts as target (bit[30] = MTYPE[3] = 1).

When I3C acts as controller:

- If the control FIFO (C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by software, or automatically by DMA, as defined by CDMAEN in the I3C_CFGR register.
- If C-FIFO is empty and a restart must be emitted with a new control word, the I3C hardware asserts the control FIFO error underrun flag (COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.
- After the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

When I3C acts as target, this register is used in register mode:

- Software writes into this register to initiate a command (IBI, controller-role or hot-join request) on the I3C bus.
- C-FIFO is disabled, and there is no DMA mode neither for control words.

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

Bit 31 **MEND**: Message end type/last message of a frame (when the I3C acts as controller)

0: this message from controller is followed by a repeated start (Sr), before another message must be emitted

1: this message from controller ends with a stop (P), being the last message of a frame

Bits 30:27 **MTYPE[3:0]**: Message type (whatever I3C acts as controller/target)

Condition: when I3C acts as I3C controller

0000: **SCL clock is forced to stop** until a next control word is executed

Bits[26:0] are ignored. On a CE1 error detection (ERRF = 1 in the I3C_EVR register and CODERR[3:0] = 0001 in the I3C_SER register) where a start/restart/stop is prevented from being generated, the software must use this message type for SCL “stuck at” recovery. Refer to *Table 528*.

0001: **header message**

Bits[26:0] are ignored. If the addressed target is not responding with an ACK to a private/direct message, as an escalation stage after a failed GETSTATUS tentative, the software must program this with EXITPTRN = 1 in the I3C_CFGR register, so that an HDR exit pattern is emitted on the bus, whatever the header is ACK-ed or NACK-ed (to avoid the target to consider that the I3C bus is in HDR mode). Refer to *Table 528* and MIPI specification about escalation handling.

If the fields of the I3C_CFGR register are programmed with EXITPTRN= 0 and RSTPTRN=1, a target reset pattern is sent on the I3C bus after the header message, whatever the header is ACK-ed or NACK-ed. This can be used to generate and broadcast a target reset pattern, especially in the case of an I3C bus topology with a single I3C target in deep sleep. Alternatively, and more generally, a target reset pattern can be encapsulated within a RSTACT CCC command (refer to *Figure 903*).

0010: **private message** (refer to *Figure 908*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred private message is:

– {S / S + 0b111_1110 + RnW = 0 + Sr/Sr+*} + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P.
– After an S (start), depending upon bit NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed, if it follows a previous message without ending by a P (stop).

0011: **direct message (second part of an I3C SDR direct CCC command)** (refer to *Figure 901*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred direct message is: Sr + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P

0100: **legacy I^2C message** (refer to *Figure 910*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit static address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred legacy I^2C message is:

– {S / S + 0b111_1110 + RnW = 0 + Sr/Sr+*} + 7-bit StaAddr + RnW + (8-bit data + T)* + Sr/P.
– After an S, depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed (if it follows a previous message without ending by a P (stop)).

Others: reserved

Condition: when I3C acts as I3C target

1000: **hot-join request** (W) (refer to *Figure 913*)

The transferred hot-join request is {S +} 0b000_0010 addr + RnW = 0.
Writing the control word initiates the hot-join request if target is allowed to do so (HJEN = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1001: **controller-role request** (W) (refer to *Figure 914*)

The transferred controller-role request is {S +} DA[6:0] + RnW = 0 (DA in the I3C_DEVR0 register)
Writing the control word initiates the controller-role request if target is allowed to do so (CREN = 1 and DAVAL = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1010: **IBI (in-band interrupt) request** (R) (refer to *Figure 912*)

Bits[15:0] (DCNT[15:0]) are the number of the IBI data payload (including the first MDB), if any.
The transferred IBI request is {S +} DA[6:0] + RnW = 1 + optional IBI data payload. Writing the control word initiates the IBI request if target is allowed to do so (IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0.register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.
When acknowledged from controller, the transmitted IBI payload data (optional, depending upon BCR2 in the I3C_BCR register) is defined by DCNT[15:0] in the I3C_CR register and I3C_IBIDR, and must be consistently programmed vs. the IBI payload data size defined by IBIP[2:0] in the I3C_IBIDR register.

Others: reserved

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:17 **ADD[6:0]**: 7-bit I3C dynamic / I^2C static target address (when I3C acts as controller)

When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message)

Bit 16 **RNW**: Read / non-write message (when I3C acts as controller)

When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to emit the RnW bit on the I3C bus.

0: write message

1: read message

Bits 15:0 **DCNT[15:0]**: Count of data to transfer during a read or write message, in bytes (whatever I3C acts as controller/target)

When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to set the number of exchanged data bytes on the bus. In case of a private or legacy I^2C read/write message, this field must be non-null.

When I3C acts as target, this field is used if MTYPE[3:0] = 1010 (IBI request) and if any IBI data payload (data to be transmitted if BCR2 = 1 in the I3C_BCR register), to set the number of bytes of the IBI data payload (1, 2, 3, or 4).

Linear encoding up to 64 Kbytes - 1

0x0000: no data to transfer

0x0001: 1 byte

0x0002: 2 bytes

...

0xFFFF: 64 Kbytes - 1 byte

45.16.2 I3C message control register [alternate] (I3C_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This write register description must be used to control the message for when the controller has to emit a CCC (whatever is the type of the CCC: for a CCC broadcast, a CCC direct, or a CCC Enter HDR).

This is the alternate description of register I3C_CR, for when MTYPE[3:0] = 0110. Else refer to *I3C message control register (I3C_CR)*.

If the control FIFO (also known as C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by the software or automatically by DMA, as defined by the CDMAEN bit in the I3C_CFGR register.

When the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

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

Bit 31 **MEND**: Message end type/last message of a frame (when I3C acts as controller)

0: this message from controller is followed by a repeated start (Sr), before another message must be emitted

1: the message from the controller ends with a stop (P), being the last message of a frame

Bits 30:27 **MTYPE[3:0]**: Message type (when I3C acts as controller)

Condition: when I3C acts as I3C controller

0110: broadcast/direct CCC command (refer to *Table 527*, *Figure 901*, *Figure 902*, *Figure 903*)

Bits[23:16] (CCC[7:0]) are the emitted 8-bit CCC code

Bits[15:0] (DCNT[15:0]) are the number of the CCC defining bytes, or CCC sub-command bytes, or CCC data bytes.

If Bit[23] = CCC[7] = 1: this is the first part of an I3C SDR direct CCC command

The transferred direct CCC command (first part) message is:

– {S / S + 0b111_1110 +RnW = 0 / Sr+*} + (direct CCC + T) + (8-bit Data + T)* + Sr
– After an S (start), depending upon NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

If Bit[23] = CCC[7] = 0: this is an I3C SDR broadcast CCC command (including specific ENTDAA, refer to *Figure 902*)

The transferred broadcast CCC command message is:

– {S / S + 0b111_1110 + RnW = 0 / Sr+*} + (broadcast CCC + T) + (8-bit Data + T)* + Sr/P
– After an S (start), depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

Others: reserved

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:16 **CCC[7:0]**: 8-bit CCC code (when I3C acts as controller)

If bit[23] = CCC[7] = 1, this is the first part of an I3C SDR direct CCC command.

If bit[23] = CCC[7] = 0, this is an I3C SDR broadcast CCC command (including ENTDAA).

Bits 15:0 **DCNT[15:0]**: Count of related data to the CCC command to transfer as CCC defining bytes, or CCC sub-command bytes, or CCC data bytes, in bytes

Linear encoding up to 64 Kbytes - 1.

0x0000: no data to transfer.

Note: Value mandatory when emitting ENTDAA broadcast CCC (refer to Figure 902).

0x0001: 1 byte

Note: Value mandatory when emitting RSTACT direct/broadcast CCC (refer to Figure 903).

0x0002: 2 bytes

...

0xFFFF: 64 Kbytes - 1 byte

45.16.3 I3C configuration register (I3C_CFGR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is used to configure:

- features that apply when the I3C acts as controller or target: RX-FIFO and TX-FIFO management (RXDMAEN, RXTHRES, RXFLUSH, TXDMAEN, TXTHRES, TXFLUSH), I3C peripheral role (CRINIT)
- dedicated features when the I3C acts as a controller: frame-based control-word triggering (TSFSET), FIFOs management (TMODE, SMODE, SFLUSH, SDMAEN, CDMAEN), and miscellaneous ones (HJACK, HKSDAEN, EXITPTRN, RSTPATRN, NOARBH, FCFDIS)

The configuration fields CRINIT, HKSDAEN can be modified only when EN = 0. This condition is respected if they are modified at the same time when EN is set to 1 (it is not necessary to set EN later on, with another write operation).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TSFSET	Res.	Res.	Res.	Res.	Res.	Res.	FCFDIS	Res.	CFLUSH	CDMA EN	TMODE	SMODE	SFLUSH	SDMA EN
	w							rw		w	rw	rw	rw	w	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TX THRES	TX FLUSH	TX DMAEN	Res.	RX THRES	RX FLUSH	RX DMAEN	HJACK	Res.	HK SDAEN	EXIT PTRN	RST PTRN	NO ARBH	CRINIT	EN
	rw	w	rw		rw	w	rw	rw		rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TSFSET**: Frame transfer set (software trigger) (when I3C acts as controller)

This bit can only be written. When I3C acts as I3C controller:

0: no action

1: setting this bit initiates a frame transfer by causing the hardware to assert the flag CFNFF in the I3C_EVR register (C-FIFO not full and a control word is needed)

Note: If this bit is not set, the other alternative for the software to initiate a frame transfer is to directly write the first control word register (I3C_CR) while C-FIFO is empty (CFEF = 1 in the I3C_EVR register). Then, if the first written control word is not tagged as a message end (MEND = 0 in the I3C_CR register), it causes the hardware to assert CFNFF.

Bits 29:24 Reserved, must be kept at reset value.

Bit 23 **FCFDIS**: FCF generation disable (when I3C acts as controller)

When the IP acts as controller, the generation of the FCF flag in the register I3C_EVR can be disabled. The software can avoid clearing this flag between two generated I3C frames (by writing 1 into CFCF in the I3C_CEVR register), especially in hardware-triggered autonomous mode.

0: enable the generation of the flag FCF in the I3C_EVR register

1: disable the generation of the flag FCF in the I3C_EVR register

Bit 22 Reserved, must be kept at reset value.

Bit 21 **CFLUSH**: C-FIFO flush (when I3C acts as controller)

This bit can only be written.

0: no action

1: flush C-FIFO

Bit 20 **CDMAEN**: C-FIFO DMA request enable (when I3C acts as controller)

When I3C acts as controller:

0: DMA mode is disabled for C-FIFO

- Software writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame
- A next control word transfer can be written by software either via polling on the flag CFNFF = 1 in the I3C_EVR register, or via interrupt notification (enabled by CFNFIE = 1 in the I3C_IER register).

1: DMA mode is enabled for C-FIFO

- DMA writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame.
- A next control word transfer is automatically written by the programmed hardware (via the asserted C-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 19 **TMODE**: Transmit mode (when I3C acts as controller)

When I3C acts as controller, this bit is used for the C-FIFO and TX-FIFO management vs. the emitted frame on the I3C bus.

0: C-FIFO and TX-FIFO are not preloaded before starting to emit a frame transfer.

A frame transfer starts as soon as the first control word is present in C-FIFO.

1: C-FIFO and TX-FIFO are first preloaded (also TX-FIFO if needed, depending on the frame format) before starting to emit a frame transfer. Refer to *Section 45.10.2* for more details.

Bit 18 **SMODE**: S-FIFO enable / status receive mode (when I3C acts as controller)

When I3C acts as controller, this bit is used to enable the FIFO for the status (S-FIFO) of the exchanged message on the I3C bus.

When I3C acts as target, this bit must be cleared.

0: S-FIFO is disabled

- Status register (I3C_SR) is used without FIFO mechanism.
- There is no SCL stalling if a new status register content is not read.
- Status register must be read before being overwritten by the hardware.
- Must have SDMAEN = 0 in the I3C_CFGR register.

1: S-FIFO is enabled.

- Each message status must be read.
- There is SCL stalling when the S-FIFO is full and a next message status must be read.
- S-FIFO overrun error is reported after the maximum SCL clock stalling time.

Bit 17 **SFLUSH**: S-FIFO flush (when I3C acts as controller)

This bit can be written and used only when I3C acts as controller.

0: no action

1: flush S-FIFO

Bit 16 **SDMAEN**: S-FIFO DMA request enable (when I3C acts as controller)

This bit must be cleared if SMODE = 0 in the I3C_CFGR register (S-FIFO is disabled). In other words, DMA mode cannot be used if S-FIFO is disabled. Then the status register I3C_SR can be read or not.

This bit can be set or cleared if SMODE = 1 (S-FIFO is enabled). In other words, status register I3C_SR must be read for each message, either by software, or via an allocated DMA channel.

0: DMA mode is disabled for reading status register I3C_SR

- SMODE = 0: software can read the I3C_SR register after a completed frame (FCF = 1 in the I3C_EVR register) or an error (ERRF = 1 in the I3C_EVR register). Via polling on these register flags or via interrupt notification (enabled by FCIE = 1 and ERRIE = 1 in the I3C_IER register).
- SMODE = 1: software must read and pop a status word from S-FIFO (read I3C_SR register) after each asserted flag SFNEF = 1. Via polling on this register flag or via interrupt notification (enabled by SFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for reading status register I3C_SR

- Must have SMODE = 1 in the I3C_CFGR register (S-FIFO enabled)
- DMA reads and pops status word(s) from S-FIFO (it reads I3C_SR register)
- Status word(s) are automatically read by the programmed hardware (via the asserted S-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TXTHRES**: TX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the TX-FIFO level, when the TXFNFF flag is set in the I3C_EVR register (and consequently if TXDMAEN = 1 when is asserted a DMA TX request).

0: 1-byte threshold

TXFNFF is set when 1 byte must be written in TX-FIFO (in I3C_TDR).

1: 1-word / 4-byte threshold

TXFNFF is set when 1 word / 4 bytes must be written in TX-FIFO (in the I3C_TDWR register). If the a number of the last transmitted data is not a multiple of 4 bytes (XDCNT[1:0] = 00 in the I3C_SR register), only the relevant 1, 2, or 3 valid LSB bytes of the last word are taken into account by the hardware, and sent on the I3C bus.

Bit 13 **TXFLUSH**: TX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

When the I3C acts as target, this bit can be used to flush the TX-FIFO on a private read if the controller has aborted the data read (driven low the T bit), and there is/are remaining data in the TX-FIFO (ABT = 1, and XDCNT[15:0] in the I3C_SR register < TGTTDCNT[15:0] in the I3C_TGTTDR register).

0: no action

1: flush TX-FIFO

Bit 12 **TXDMAEN**: TX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for TX-FIFO

- Software writes and pushes a data byte/word into TX-FIFO (writes I3C_TDR or I3C_TDWR register), to be transmitted over the I3C bus.
- A next data byte/word must be written by the software either via polling on the flag TXFNFF = 1 or via interrupt notification (enabled by TXFNFIE = 1).

1: DMA mode is enabled for TX-FIFO

- DMA writes and pushes data byte(s)/word(s) into TX-FIFO (writes I3C_TDR or I3C_TDWR register).
- A next data byte/word transfer is automatically pushed by the programmed hardware (via the asserted TX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 11 Reserved, must be kept at reset value.

Bit 10 **RXTHRES**: RX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the RX-FIFO level, when the RXFNEF flag in the I3C_EVR register is set (and consequently if RXDMAEN = 1 when is asserted a DMA RX request).

0: 1-byte threshold

RXFNEF is set when 1 byte must be read in RX-FIFO (in the I3C_RDR register).

1: 1-word/4-bytes threshold

RXFNEF is set when 1 word / 4 bytes is/are to be read in RX-FIFO (in I3C_RDWR). In the case of a number of last received data being not a multiple of 4 bytes, only the relevant 1, 2 or 3 valid LSB bytes of the last word are to be considered by the software. The number of effective received data bytes is reported by XDCNT[15:0] in the I3C_SR register.

Bit 9 **RXFLUSH**: RX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

0: no action

1: flush RX-FIFO

Bit 8 **RXDMAEN**: RX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for RX-FIFO

- Software reads and pops a data byte/word from RX-FIFO (it reads I3C_RDR or I3C_RDWR register).
- A next data byte/word must be read by the software either via polling flag RXFNEF = 1 in the I3C_EVR register, or via interrupt notification (enabled by RXFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for RX-FIFO

- DMA reads and pops data byte(s)/word(s) from RX-FIFO (reads I3C_RDR or I3C_RDWR register).
- A next data byte/word is automatically read by the programmed hardware (via the asserted RX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 7 **HJACK**: Hot-join request acknowledge (when I3C acts as a controller)

0: hot-join request is not acknowledged

After the NACK, the controller continues as initially programmed (the hot-joining target is aware of the NACK and must emit another hot-join request later on).

1: hot-join request is acknowledged

After the ACK, the controller continues as initially programmed. The software is notified by the HJ interrupt (flag HJF is set in the I3C_EVR register), and must initiate the ENTDAA sequence later on, potentially preventing other hot-join requests with a disable target events command (DISEC, with DISHJ = 1).

Bit 6 Reserved, must be kept at reset value.

Bit 5 **HKSDAEN:** High-keeper enable on SDA line (when I3C acts as a controller)

0: High-keeper is disabled

1: High-keeper is enabled, and the weak pull-up is effective on the T bit, instead of the open-drain class pull-up.

Note: This bit can be modified only when EN = 0 in the I3C_CFGR register.

Bit 4 **EXITPTRN:** HDR exit pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: HDR exit pattern is not sent after the issued message header (MTYPE[3:0] = 0001 in the I3C_CR register). This is used to send the header, to test ownership of the bus when there is a suspicion of a problem after controller-role hand-off (new controller did not assert its controller-role by accessing the previous one in less than the delay defined by the activity state).

1: HDR exit pattern is sent after the issued message header (MTYPE[3:0] = 0001). This is used on a controller error detection and escalation handling, in case of a not responding target to a private message or a direct read CCC.

The HDR exit pattern is sent whatever the message header {S/Sr + 0x7E addr + W} is ACK-ed or NACK-ed..

Bit 3 **RSTPTRN:** HDR reset pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: standard stop emitted at the end of a frame

1: HDR reset pattern is inserted before the stop of any emitted frame that includes a RSTACT CCC command

Additionally, if RSTPTRN = 1 and EXITPTRN = 0, a reset pattern is sent after the issued message header (MTYPE[3:0] = 0001 in the I3C_CR register), whatever the message header is ACK-ed or NACK-ed.

Bit 2 **NOARBH**: No arbitrable header after a start (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: An arbitrable header (0b111_1110 + RnW = 0) is emitted after a start and before a legacy I^2C message or an I3C SDR private read/write message (default).

1: No arbitrable header

- The target address is emitted directly after a start in case of a legacy I^2C message or an I3C SDR private read/write message.

- This is a more performing option (when the emission of the 0x7E arbitrable header is useless), but must be used only when the controller is sure that the addressed target device cannot emit concurrently an IBI or a controller-role request (to ensure no misinterpretation and no potential conflict between the address emitted by the controller in open-drain mode and the same address a target device can emit after a start, for IBI or MR).

Bit 1 **CRINIT**: Initial controller/target role

This bit can be modified only when EN = 0 in the I3C_CFGR register.

0: target role

Once enabled by setting EN = 1, the I3C peripheral initially acts as a target. I3C does not drive SCL line and does not enable SDA pull-up, until it eventually acquires the controller role.

1: controller role

Once enabled by setting EN = 1, the I3C peripheral initially acts as a controller. It has the I3C controller role, so drives SCL line and enables SDA pull-up, until it eventually offers the controller role to an I3C secondary controller.

Bit 0 **EN**: I3C enable (whatever I3C acts as controller/target)

0: I3C is disabled

- Except registers, the I3C peripheral is under reset (partial reset).
- Before clearing EN, when I3C acts as a controller, all the possible target requests must be disabled using DISEC CCC.
- When I3C acts as a target, software must not disable the I3C, unless a partial reset is needed.

1: I3C is enabled

In this state, some register fields cannot be modified (like CRINIT, HKSDAEN for the I3C_CFGR)

45.16.4 I3C receive data byte register (I3C_RDR)

Address offset: 0x010

Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a byte-based read access (RXTHRES = 0 in the I3C_CFGR register). Then:

- On read, I3C_RDR returns a 32-bit data word, with the received data byte in LSB position, and other reserved bits read as 0.
- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must wait to be notified that a next received data byte must be read via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received byte of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.
- If the RX-FIFO is full, a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by error flag ERRF = 1 in the I3C_EVR register and by a data overrun flag DOVRF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDB0[7:0]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **RDB0[7:0]**: 8-bit received data on I3C bus.

45.16.5 I3C receive data word register (I3C_RDWR)

Address offset: 0x014

Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a 32-bit word-based read access (RXTHRES = 1 in the I3C_CFGR register). Then:

- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must first wait to be notified that must be read a next received data word via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received word/byte(s) of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.
- If the RX-FIFO holds less than four bytes, a read on I3C_RDWR returns a data word padded with null byte(s): the available byte(s) in LSB position(s) is (are) padded with zero byte(s) in MSB position(s).
- If the RX-FIFO is full and a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data overrun DOVR = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDB3[7:0]								RDB2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDB1[7:0]								RDB0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **RDB3[7:0]**: 8-bit received data (latest byte on I3C bus).

Bits 23:16 **RDB2[7:0]**: 8-bit received data (next byte after RDB1 on I3C bus).

Bits 15:8 **RDB1[7:0]**: 8-bit received data (next byte after RDB0 on I3C bus).

Bits 7:0 **RDB0[7:0]**: 8-bit received data (earliest byte on I3C bus).

45.16.6 I3C transmit data byte register (I3C_TDR)

Address offset: 0x018

Reset value: 0x0000 0000

This register is used to write data bytes to be transmitted over the I3C bus.

This register implements a byte-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 0 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 0 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 0: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last transmitted byte of a given message to be written is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 0 and if PRELOAD = 1 in the I3C_TGTTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to I3C_TDR register, according to TGTTDCNT[15:0] in the I3C_TGTDR register.
- When TXDMAEN = 0 and if TXTHRES = 0:
  - when PRELOAD = 1 in the I3C_TGTTDR register: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non full flag (TXFNFF = 1) or via the corresponding interrupt, if enabled. The last transmitted byte of the initially programmed TGTTDCNT[15:0] is also marked with TXLASTF = 1 in the I3C_EVR register.
  - when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to 8 next data bytes (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. If the software needs to write another data byte, it must wait for the TX-FIFO be empty (TXFEF = 1), and then write this data byte to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 (and corresponding interrupt if enabled) in the I3C_SER register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDB0[7:0]							
								w	w	w	w	w	w	w	w

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

Bits 7:0 **TDB0[7:0]**: 8-bit data to transmit on I3C bus.

45.16.7 I3C transmit data word register (I3C_TDWR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to write 32-bit data words to be transmitted over the I3C bus.

This register implements a word-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 1 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 1 in the I3C_CFGR register, programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 1: before a write, the software must wait to be notified that next data word/byte(s) must be written via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. Last transmitted word/byte(s) of a given message to be written in TX-FIFO is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 1 and if PRELOAD = 1 in the I3C_TGTTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to the I3C_TDWR register, as per TGTTDCNT[15:0] in the I3C_TGTDR register.
- When TXDMAEN = 0 and if TXTHRES = 1:
  - when PRELOAD = 1: before a write, the software must wait to be notified that must be written a next data word via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt, if enabled. Last transmitted word of the initially programmed TGTTDCNT[15:0] in the I3C_TGTTDR register is also marked with TXLASTF = 1 in the I3C_EVR register.
  - when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to two next data words (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. If the software needs to write another data word, it must wait for TX-FIFO to be empty (TXFEF = 1), and then write the next data word to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller/target cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDB3[7:0]								TDB2[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDB1[7:0]								TDB0[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:24 **TDB3[7:0]**: 8-bit transmit data (latest byte on I3C bus).

Bits 23:16 **TDB2[7:0]**: 8-bit transmit data (next byte after TDB1[7:0] on I3C bus).

Bits 15:8 **TDB1[7:0]**: 8-bit transmit data (next byte after TDB0[7:0] on I3C bus).

Bits 7:0 **TDB0[7:0]**: 8-bit transmit data (earliest byte on I3C bus)

45.16.8 I3C IBI payload data register (I3C_IBIDR)

Address offset: 0x020

Reset value: 0x0000 0000

This register is used for the IBI payload data.

When I3C acts as target:

- if BCR2 = 0 in the I3C_BCR register, this register is useless.
- if BCR2 = 1, this register must be written by software
  - to be emitted on the I3C bus as the IBI payload data, after that the IBI request (MTYPE[3:0] = 1010 in the I3C_CR register) is acknowledged by the controller; with the IBI data payload size defined by DCNT[15:0] in the I3C_CR register.
  - Maximum (static) payload data size is given by IBIP[2:0] in the I3C_MAXRLR register. it can be 1, 2, 3 or 4 bytes.
  - DCNT[15:0] must be set between 1 (for the mandatory data byte MDB[7:0]) and the maximum IBIP[2:0].

When I3C acts as controller: if IBIACK= 1 in the I3C_DEVRx register, once it acknowledges on the I3C bus the IBI request from the target x:

- if IBIDEN = 0 (BCR[2] = 0 received from target x) in the I3C_DEVRx register, this register is useless.
- if IBIDEN = 1 (BCR[2] = 1 received from target x):
  - This register is internally written by hardware from the IBI payload data received on the I3C bus (including the first mandatory data byte MDB[7:0]).
  - When the last byte of the payload is received in I3C_IBIDR (when target drives T-bit = 0), IBIF flag is set (and corresponding interrupt if enabled) in the I3C_EVR register.
  - Then, the software can identify the target x via the received and logged 7-bit address in RADD[6:0] in the I3C_RMR register.
  - The software can read this register and interpret byte(s) according to the number of bytes effectively received and logged in IBIRDCNT[2:0] in the I3C_RMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IBIDB3[7:0]								IBIDB2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIDB1[7:0]								IBIDB0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **IBIDB3[7:0]**: 8-bit IBI payload data (latest byte on I3C bus).

Bits 23:16 **IBIDB2[7:0]**: 8-bit IBI payload data (next byte on I3C bus after IBIDB1[7:0]).

Bits 15:8 **IBIDB1[7:0]**: 8-bit IBI payload data (next byte on I3C bus after IBIDB0[7:0]).

Bits 7:0 **IBIDB0[7:0]**: 8-bit IBI payload data (earliest byte on I3C bus, MDB[7:0] mandatory data byte).

45.16.9 I3C target transmit configuration register (I3C_TGTTDR)

Address offset: 0x024

Reset value: 0x0000 0000

When I3C acts as target, this register must be used to preload a number of data bytes in the TX-FIFO, so that they are ready to be transmitted on the I3C bus, when they are accepted/acknowledged by the controller, whatever the DMA mode is used or not (independently from TXDMAEN in the I3C_CFGR register).

When I3C acts as target, alternatively, if the number of data bytes to be transmitted is less than or equal to the TX-FIFO size (8 bytes), the software can directly use and write byte(s) into the I3C_TDR register (or I3C_TDWR register, depending upon TXTHRES in the I3C_CFGR register), via polling on the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding enabled interrupt.

In any case, since an I3C target is unable to stretch/stall the SCL line, the software can identify a TX-FIFO underrun via DOR = 1 in the I3C_EVR register.

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

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **PRELOAD**: Preload of the TX-FIFO (when I3C is configured as target)

This bit must be written and asserted by software in the same access when is written and defined the number of bytes to preload into the TX-FIFO and to transmit.

This bit is cleared by hardware when all the data bytes to transmit are loaded into the TX-FIFO.

0: no TX-FIFO preload

1: TX-FIFO preload

Bits 15:0 **TGTTDCNT[15:0]**: Transmit data counter, in bytes (when I3C is configured as target)

This bitfield must be written by software in the same access when is asserted PRELOAD, in order to define the number of bytes to preload and to transmit.

This bitfield is updated by hardware and reports, when read, the remaining number of bytes to be loaded into the TX-FIFO.

45.16.10 I3C status register (I3C_SR)

Address offset: 0x030

Reset value: 0x0000 0000

This register is used to read the status about the exchanged message on the I3C bus:

- in FIFO mode: when the I3C acts as controller and if S-FIFO is enabled via SMODE = 1 in the I3C_CFGR register:
  - Software is notified via SFNEF = 1 in the I3C_EVR register if there is a status register to be read (and corresponding interrupt if enabled), when not in DMA mode (SDMAEN = 0)
  - In DMA mode (SDMAEN = 1), programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
  - Software is notified via COVR = 1 in the I3C_SER register on an S-FIFO overflow (and ERRF = 1 in the I3C_EVR register and corresponding interrupt if enabled)
- in register mode: if SMODE = 0 in the I3C_CFGR register
  - Software can use the flags FCF in the I3C_SER register and ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled) to read this register
  - This register can be overwritten by hardware on a new message completion, without any notification.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MID[7:0]								Res.	Res.	Res.	Res.	Res.	DIR	ABT	Res.
r	r	r	r	r	r	r	r						r	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XDCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **MID[7:0]**: Message identifier/counter of a given frame (when the I3C acts as controller)

When I3C acts as controller, this fbitield identifies the control word message (I3C_CR) to whom the I3C_SR status register refers.

First message of a frame is identified with MID[7:0] = 0.

This bitfield is incremented (by hardware) on the completion of a new message control word (I3C_CR) over I3C bus. This field is reset for every new frame start.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **DIR**: Message direction

Whatever the I3C acts as controller or target, this bit indicates the direction of the related message on the I3C bus

0: write

1: read

Note: ENTDAA CCC is considered as a write command.

Bit 17 **ABT**: A private read message is ended prematurely by the target (when the I3C acts as controller)

When the I3C acts as controller, this bit indicates if the private read data transmitted by the target early terminates (the target drives T bit low earlier vs. what the controller expects in terms of programmed number of read data bytes DCNT[15:0] in the I3C_CR register).

0: no early completion from the target

1: early completion from the target

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **XDCNT[15:0]**: Data counter

Condition: during the dynamic address assignment process (ENTDAA CCC)

- When the I3C acts as controller: number of targets detected on the bus
- When the I3C acts as target: number of transmitted bytes

Condition: for other transfers, during the message

- Whatever the I3C acts as controller or target: number of data bytes read from or transmitted on the I3C bus during the message

45.16.11 I3C status error register (I3C_SER)

Address offset: 0x034

Reset value: 0x0000 0000

This read register is used to get more information about the error when an error is raised by hardware and notified to the software via the error flag ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled).

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

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DERR**: Data error (when the I3C acts as controller)

0: no detected error

1: controller detected a data error during the controller-role hand-off procedure (GETACCCR CCC, formerly known as GETACCMST) when the received target address or/and the parity bit do no match. Active controller keeps controller-role.

Bit 9 **DNACK**: Data not acknowledged (when the I3C acts as controller)

0: no detected error

1: controller detected that a data byte is not acknowledged by a target, either during:

i) a legacy I2C write transfer

ii) the second trial when sending dynamic address during ENTDAA procedure

Bit 8 **ANACK**: Address not acknowledged (when the I3C is configured as controller)

0: no detected error

1: controller detected that the static/dynamic address was not acknowledged by a target, either during:

i) a legacy I2C read/write transfer

ii) a direct CCC write transfer

iii) the second trial of a direct CCC read transfer

iv) a private read/write transfer

Bit 7 **COVR**: C-FIFO underrun or S-FIFO overrun (when the I3C acts as controller)

0: no detected error

1: controller detected either:

i) a C-FIFO underrun: control FIFO is empty and a restart must be emitted

ii) an S-FIFO overrun: S-FIFO is full and a new message ends

Bit 6 **DOVR**: RX-FIFO overrun or TX-FIFO underrun

0: no detected error

1: whatever controller or target, hardware detected either:

i) a TX-FIFO underrun: TX-FIFO is empty and a write data byte must be transmitted

ii) a RX-FIFO overrun: RX-FIFO is full and a new data byte is received

Bit 5 **STALL**: SCL stall error (when the I3C acts as target)

0: no detected error

1: target detected that SCL was stable for more than 125 µs during an I3C SDR data read (during a direct CCC read, a private read, or an IB)

Bit 4 **PERR**: Protocol error

0: no detected error

1: whatever controller or target, hardware detected a protocol error, as detailed in CODERR[3:0]

Bits 3:0 **CODERR[3:0]**: Protocol error code/type
0000: CE0 error (transaction after sending CCC):
controller detected an illegally formatted CCC
0001: CE1 error (monitoring error):
controller detected that transmitted data on the bus is different from expected
0010: CE2 error (no response to broadcast address):
controller detected a not acknowledged broadcast address (0b111_1110)
0011: CE3 error (failed controller-role hand-off):
controller detected the new controller did not drive bus after controller-role hand-off
1000: TE0 error (invalid broadcast address 0b111_1110 + W):
target detected an invalid broadcast address 0b111_1110 + W
1001: TE1 error (CCC code):
target detected a parity error on a CCC code via a parity check (vs. T bit)
1010: TE2 error (write data):
target detected a parity error on a write data via a parity check (vs. T bit)
1011: TE3 error (assigned address during dynamic address arbitration):
target detected a parity error on the assigned address during dynamic address arbitration via a parity check (vs. PAR bit)
1100: TE4 error (0b111_1110 + R missing after Sr during dynamic address arbitration):
target detected a 0b111_1110 + R missing after Sr during dynamic address arbitration
1101: TE5 error (transaction after detecting CCC):
target detected an illegally formatted CCC
1110: TE6 error (monitoring error):
target detected that transmitted data on the bus is different from expected
others: reserved

45.16.12 I3C received message register (I3C_RMR)

Address offset: 0x040

Reset value: 0x0000 0000

When the I3C acts as controller, this read register is used to log the received target address, and the IBI received payload data size.

When the I3C acts as target, this read register is used to log the received CCC code

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **RADD[6:0]**: Received target address (when the I3C is configured as controller)
When the I3C is configured as controller, this field logs the received dynamic address from the target during acknowledged IBI or controller-role request.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 **RCODE[7:0]**: Received CCC code (when the I3C is configured as target)
When the I3C is configured as target, this field logs the received CCC code.

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

Bits 2:0 **IBIRDCNT[2:0]**: IBI received payload data count (when the I3C is configured as controller)
When the I3C is configured as controller, this field logs the number of data bytes effectively received in the I3C_IBIDR register.

45.16.13 I3C event register (I3C_EVR)

Address offset: 0x050

Reset value: 0x0000 0003

This is a read register, used for reporting event flags.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPF	DEFF	INT UPDF	AS UPDF	RSTF	MRL UPDF	MWL UPDF	DA UPDF	STAF	GETF	WKPF	Res.	HJF	CR UPDF	CRF	IBI ENDF
r	r	r	r	r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIF	Res.	Res.	Res.	ERRF	RXTGT ENDF	FCF	Res.	RX LASTF	TX LASTF	RX FNEF	TX FNFF	SFNEF	CFNFF	TXFEF	CFEF
r				r	r	r		r	r	r	r	r	r	r	r

Bit 31 **GRPF**: Group addressing flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller-capable), this flag is asserted by hardware to indicate that the broadcast DEFGRPA CCC (define list of group addresses) has been received. Then, software can store the received data for when getting controller role.
The flag is cleared when software writes 1 into the corresponding CGRPF bit in the I3C_CR register.

Bit 30 **DEFF**: DEFTGTS flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller capable), this flag is asserted by hardware to indicate that the broadcast DEFTGTS CCC (define list of targets) has been received. Then, software can store the received data for when getting the controller role.
The flag is cleared when software writes 1 into the corresponding CDEFF bit in the I3C_CEVR register.

Bit 29 **INTUPDF**: Interrupt/controller-role/hot-join update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENEC/DISEC CCC (enable/disable target events) has been received, where a target event is either an interrupt/IBI request, a controller-role request, or an hot-join request. Then, software must read respectively IBIEN, CREN, or HJEN in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CINTUPDF bit in the I3C_CEVR register.

Bit 28 **ASUPDF**: Activity state update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENTASx CCC (with x = 0...3) has been received. Then, software must read AS[1:0] in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CASUPDF bit in the I3C_CEVR register.

Bit 27 **RSTF**: Reset pattern flag (when the I3C acts as target)

When I3C acts as target, this flag is asserted by hardware to indicate that a reset pattern has been detected (14 SDA transitions while SCL is low, followed by repeated start, then stop).

Then, when not in Stop mode, software must read RSTACT[1:0] and RSTVAL in the I3C_DEVR0 register, to know the required reset level.

- If RSTVAL = 1: when the RSTF is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.
- If RSTVAL = 0: when the RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.

When in Stop mode, the corresponding interrupt can be used to wake up the device.

The flag is cleared when software writes 1 into the corresponding CRSTF bit in the I3C_CEVR register.

Bit 26 **MRLUPDF**: Maximum read length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMRL CCC (set max read length) has been received. Then, software must read MRL[15:0] in the I3C_MAXRLR register to get the maximum read length value.

The flag is cleared when software writes 1 into the corresponding CMRLUPDF bit in the I3C_CEVR register.

Bit 25 **MWLUPDF**: Maximum write length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMWL CCC (set max write length) has been received. Then, software must read MWL[15:0] in the I3C_MAXRLR register to get the maximum write length value.

The flag is cleared when software writes 1 into the corresponding CMWLUPDF bit in the I3C_CEVR register.

Bit 24 **DAUPDF**: Dynamic address update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a dynamic address update has been received via any of the broadcast ENTDAA, RSTDAA and direct SETNEWDA CCC. Then, software must read DA[6:0] and DAVAL in the I3C_DEVR0 register to get the dynamic address update.

The flag is cleared when software writes 1 into the corresponding CDAUPDF bit in the I3C_CEVR register.

Bit 23 **STAF**: Get status flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct GETSTATUS CCC of format 1 (without defining byte or with defining byte TGTSTAT) has been received.

The flag is cleared when software writes 1 into the corresponding CSTAF bit in the I3C_CEVR register.

Bit 22 **GETF**: Get flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that any direct CCC of get type (GET*** CCC) except the GETSTATUS of format 1 (but including GETSTATUS of format 2) has been received.

The flag is cleared when software writes 1 into the corresponding CGETF bit in the I3C_CEVR register.

Bit 21 **WKPF**: Wake-up/missed start flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a start has been detected (an SDA falling edge followed by an SCL falling edge) but on the next SCL falling edge, the I3C kernel clock is (still) gated. Thus an I3C bus transaction may have been lost by the target.

The corresponding interrupt can be used to wake up the device from a low power (Sleep or Stop) mode.

The flag is cleared when software writes 1 into the corresponding CWKPF bit in the I3C_CEVR register.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJF**: Hot-join flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that an hot join request has been received.

The flag is cleared when software writes 1 into the corresponding CHJF bit in the I3C_CEVR register.

Bit 18 **CRUPDF**: Controller-role update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that it has now gained the controller role after the completed controller-role hand-off procedure.

The flag is cleared when software writes 1 into the corresponding CCRUPDF bit in the I3C_CEVR register.

Bit 17 **CRF**: Controller-role request flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a controller-role request has been acknowledged and completed (by hardware). The software must then issue a GETACCCR CCC (get accept controller role) for the controller-role hand-off procedure.

The flag is cleared when software writes 1 into the corresponding CCRF bit in the I3C_CEVR register.

Bit 16 **IBIENDF**: IBI end flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that an IBI transfer has been received and completed (IBI acknowledged and IBI data bytes read by controller if any).

The flag is cleared when software writes 1 into the corresponding CIBIENDF bit in the I3C_CEVR register.

Bit 15 **IBIF**: IBI flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that an IBI request has been received.

The flag is cleared when software writes 1 into the corresponding CIBIF bit in the I3C_CEVR register.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRF**: Flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that an error occurred.Then, software must read I3C_SER to get the error type.

The flag is cleared when software writes 1 into the corresponding CERRF bit in the I3C_CEVR register.

Bit 10 **RXTGTENDF**: Target-initiated read end flag (when the I3C acts as controller)

When the I3C acts as controller, and only if the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register), this flag is asserted by hardware to indicate that the target has prematurely ended a read transfer. Then, software must read the status register I3C_SR to check information related to the last message and get the number of received data bytes on the prematurely read transfer (XDCNT in the I3C_SR register).

The flag is cleared when software writes 1 into the corresponding CRXTGTENDF bit in the I3C_CEVR register.

Bit 9 **FCF**: Frame complete flag (whatever the I3C acts as controller/target)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a frame has been (normally) completed on the I3C bus, for example, when a stop is issued.

When the I3C acts as target, this flag is asserted by hardware to indicate that a message addressed to/by this target has been (normally) completed on the I3C bus, for example, when a next stop or repeated start is then issued by the controller.

The flag is cleared when software writes 1 into the corresponding CFCF bit in the I3C_CEVR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **RXLASTF**: Last read data byte/word flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that the last data byte/word (depending upon RXTHRES in the I3C_CFGR register) of a message must be read from the RX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is read.

Bit 6 **TXLASTF**: Last written data byte/word flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the last data byte/word (depending upon TXTHRES in the I3C_CFGR register) of a message must be written to the TX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is written.

Bit 5 **RXFNEF**: RX-FIFO not empty flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte must/must not be read from the RX-FIFO.

Note: The software must wait for RXFNEF = 1 (by polling or via the enabled interrupt) before reading from RX-FIFO (reading from I3C_RDR or I3C_RDWR, depending upon RXTHRES).

Bit 4 **TXFNFF**: TX-FIFO not full flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte/word must/must not be written to the TX-FIFO.

Note: The software must wait for TXFNFF = 1 (by polling or via the enabled interrupt) before writing to TX-FIFO (writing to I3C_TDR or I3C_TDWR, depending upon TXTHRES).

Note: When the I3C acts as target, if the software intends to use the TXFNFF flag for writing into I3C_TDR/I3C_TDWR, it must have configured and set the TX-FIFO preload (write PRELOAD in the I3C_TGTTDR register).

Bit 3 **SFNEF**: S-FIFO not empty flag (when the I3C acts as controller)

When the I3C acts as controller, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), this flag is asserted by hardware to indicate that a status word must be read from the S-FIFO. The flag is de-asserted by hardware to indicate that a status word is not to be read from the S-FIFO.

Bit 2 **CFNFF**: C-FIFO not full flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a control word must be written to the C-FIFO. The flag is de-asserted by hardware to indicate that a control word is not to be written to the C-FIFO.

Note: The software must wait for CFNFF = 1 (by polling or via the enabled interrupt) before writing to C-FIFO (writing to I3C_CR).

Bit 1 **TXFEF**: TX-FIFO empty flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the TX-FIFO is empty.

This flag is de-asserted by hardware to indicate that the TX-FIFO is not empty.

Bit 0 **CFEF**: C-FIFO empty flag (whatever the I3C acts as controller)

This flag is asserted by hardware to indicate that the C-FIFO is empty when controller, and that the I3C_CR register contains no control word (none IBI/CR/HJ request) when target.

This flag is de-asserted by hardware to indicate that the C-FIFO is not empty when controller, and that the I3C_CR register contains one control word (a pending IBI/CR/HJ request) when target.

Note: When the I3C acts as controller, if the C-FIFO and TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), the software must wait for TXFEF = 1 and CFEF = 1 before starting a new frame transfer.

45.16.14 I3C interrupt enable register (I3C_IER)

Address offset: 0x054

Reset value: 0x0000 0000

This register is used to enable/disable, at bit level, an interrupt, for each of the following event/flag.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPIE	DEFIE	INTUPDIE	ASUPDIE	RSTIE	MRLUPDIE	MWLUPDIE	DAUPDIE	STAIE	GETIE	WKPIE	Res.	HJIE	CRUPDIE	CRIE	IBIENDIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIIE	Res.	Res.	Res.	ERRIE	RXTGTENDIE	FCIE	Res.	Res.	Res.	RXFNEIE	TXFNFIE	SFNEIE	CFNFIE	Res.	Res.
rw				rw	rw	rw				rw	rw	rw	rw		

Bit 31 **GRPIE**: DEFGRPA CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 30 **DEFIE**: DEFTGTS CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 29 **INTUPDIE**: ENEC/DISEC CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 28 **ASUPDIE**: ENTASx CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 27 **RSTIE**: reset pattern interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 26 **MRLUPDIE**: SETMRL CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 25 **MWLUPDIE**: SETMWL CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 24 **DAUPDIE**: ENTDAA/RSTDAA/SETNEWDA CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 23 **STAIE**: format 1 GETSTATUS CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 22 **GETIE**: GETxxx CCC interrupt enable (except GETSTATUS of format 1) (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 21 **WKPIE**: Wake-up interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJIE**: Hot-join interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 18 **CRUPDIE**: Controller-role update interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 17 **CRIE**: Controller-role request interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 16 **IBIENDIE**: IBI end interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 15 **IBIIE**: IBI request interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRIE**: error interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 10 **RXTGTENDIE**: target-initiated read end interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 9 **FCIE**: frame complete interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **RXFNEIE**: RX-FIFO not empty interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 4 **TXFNFIE**: TX-FIFO not full interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 3 **SFNEIE**: S-FIFO not empty interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

Bit 2 **CFNFIE**: C-FIFO not full interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

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

45.16.15 I3C clear event register (I3C_CEVR)

Address offset: 0x058

Reset value: 0x0000 0000

This write register is used to clear individually, at bit level, the corresponding event flag of the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CGRPF	CDEFF	CINTUPDF	CASUPDF	CRSTF	CMRLUPDF	CMWLUPDF	CDAUPDF	CSTAF	CGETF	CWKPF	Res.	CHJF	CCRUPDF	CCRF	CIBIENDF
w	w	w	w	w	w	w	w	w	w	w		w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIBIF	Res.	Res.	Res.	CERRF	CRXTGTENDF	CFCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w				w	w	w									

Bit 31 **CGRPF**: Clear DEFGRPA CCC flag (when the I3C acts as target)
0: no effect
1: clear GRPF

Bit 30 **CDEFF**: Clear DEFTGTS CCC flag (when the I3C acts as target)
0: no effect
1: clear DEFF

Bit 29 **CINTUPDF**: Clear ENEC/DISEC CCC flag (when the I3C acts as target)
0: no effect
1: clear CINTUPDF

Bit 28 **CASUPDF**: Clear ENTASx CCC flag (when the I3C acts as target)
0: no effect
1: clear ASUPDF

Bit 27 **CRSTF**: Clear reset pattern flag (when the I3C acts as target)
0: no effect
1: clear RSTF

Bit 26 **CMRLUPDF**: Clear SETMRL CCC flag (when the I3C acts as target)
0: no effect
1: clear MRLUPDF

Bit 25 **CMWLUPDF**: Clear SETMWL CCC flag (when the I3C acts as target)
0: no effect
1: clear MWLUPDF

Bit 24 **CDAUPDF**: Clear ENTDAA/RSTDAA/SETNEWDA CCC flag (when the I3C acts as target)
0: no effect
1: clear DAUPDF

Bit 23 **CSTAF**: Clear format 1 GETSTATUS CCC flag (when the I3C acts as target)
0: no effect
1: clear STAF

Bit 22 **CGETF**: Clear GETxxx CCC flag (except GETSTATUS of format 1) (when the I3C acts as target)
0: no effect
1: clear GETF

Bit 21 **CWKPF**: Clear wake-up flag (when the I3C acts as target)
0: no effect
1: clear WKPF

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CHJF**: Clear hot-join flag (when the I3C acts as controller)
0: no effect
1: clear HJF

Bit 18 **CCRUPDF**: Clear controller-role update flag (when the I3C acts as target)
0: no effect
1: clear CRUPDF

Bit 17 **CCRF**: Clear controller-role request flag (when the I3C acts as controller)
0: no effect
1: clear CRF

Bit 16 **CIBIENDF**: Clear IBI end flag (when the I3C acts as target)
0: no effect
1: clear IBIENDF

Bit 15 **CIBIF**: Clear IBI request flag (when the I3C acts as controller)
0: no effect
1: clear IBIF

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **CERRF**: Clear error flag (whatever the I3C acts as controller/target)
0: no effect
1: clear ERRF

Bit 10 **CRXTGTENDF**: Clear target-initiated read end flag (when the I3C acts as controller)
0: no effect
1: clear RXTGTENDF

Bit 9 **CFCF**: Clear frame complete flag (whatever the I3C acts as controller/target)
0: no effect
1: clear FCF

Bits 8:0 Reserved, must be kept at reset value.

45.16.16 I3C masked interrupt status register (I3C_MISR)

Address offset: 0x05C

Reset value: 0x0000 0000

This is a read register, containing the masked interrupt status for each individual event flag bit. Each bit is the logical AND of the corresponding interrupt enable bit of the I3C_IER register and the event flag bit of the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPMIS	DEFMIS	INTUPDMIS	ASUPDMIS	RSTMIS	MRLUPDMIS	MWLUPDMIS	DAUPDMIS	STAMIS	GETMIS	WKPMIS	Res.	HJMIS	CRUPDMIS	CRMIS	IBIENDMIS
r	r	r	r	r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIMIS	Res.	Res.	Res.	ERRMIS	RXTGTENDMIS	FCMIS	Res.	Res.	Res.	RXFNEMIS	TXFNFMIS	SFNEMIS	CFNFMIS	Res.	Res.
r				r	r	r				r	r	r	r		

Bit 31 **GRPMIS**: DEFGRPA CCC masked interrupt status (when the I3C acts as target)

Bit 30 **DEFMIS**: DEFTGTS CCC masked interrupt status (when the I3C acts as target)

Bit 29 **INTUPDMIS**: ENEC/DISEC CCC masked interrupt status (when the I3C acts as target)

Bit 28 **ASUPDMIS**: ENTASx CCC masked interrupt status (when the I3C acts as target)

Bit 27 **RSTMIS**: reset pattern masked interrupt status (when the I3C acts as target)

Bit 26 **MRLUPDMIS**: SETMRL CCC masked interrupt status (when the I3C acts as target)

Bit 25 **MWLUPDMIS**: SETMWL CCC masked interrupt status (when the I3C acts as target)

Bit 24 **DAUPDMIS**: ENTDAA/RSTDAA/SETNEWDA CCC masked interrupt status (when the I3C acts as target)

Bit 23 **STAMIS**: format 1 GETSTATUS CCC masked interrupt status (when the I3C acts as target)

Bit 22 **GETMIS**: GETxxx CCC masked interrupt status, except GETSTATUS of format 1 (when the I3C acts as target)

Bit 21 **WKPMIS**: Wake-up/missed start masked interrupt status (when the I3C acts as target)

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJMIS**: Hot-join masked interrupt status (when the I3C acts as controller)

Bit 18 **CRUPDMIS**: Controller-role update masked interrupt status (when the I3C acts as target)

Bit 17 **CRMIS**: Controller-role request masked interrupt status (when the I3C acts as controller)

Bit 16 **IBIENDMIS**: IBI end masked interrupt status (when the I3C acts as target)

Bit 15 **IBIMIS**: IBI request masked interrupt status (when the I3C acts as controller)

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRMIS**: error masked interrupt status (whatever the I3C acts as controller/target)

Bit 10 **RXTGTENDMIS**: target-initiated read end masked interrupt status (when the I3C acts as controller)

Bit 9 **FCMIS**: frame complete masked interrupt status (whatever the I3C acts as controller/target)

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **RXFNEMIS**: RX-FIFO not empty masked interrupt status (whatever the I3C acts as controller/target)

Bit 4 **TXFNFMIS**: TX-FIFO not full masked interrupt status (whatever the I3C acts as controller/target)

Bit 3 **SFNEMIS**: S-FIFO not empty masked interrupt status (when the I3C acts as controller)

Bit 2 **CFNFMIS**: C-FIFO not full masked interrupt status (when the I3C acts as controller)

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

45.16.17 I3C own device characteristics register (I3C_DEVR0)

Address offset: 0x060

Reset value: 0x0000 0000

When the I3C peripheral acts as target, this register is used to write or read its own device characteristics.

When the I3C peripheral acts as controller, the field DA[6:0] is used to write and store its own dynamic address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTVAL	RSTACT[1:0]		AS[1:0]		HJEN	Res.	CREN	IBIEN
							r	r	r	r	r	rw		rw	rw

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

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RSTVAL**: Reset action is valid (when the I3C acts as target)

This bit is asserted by hardware to indicate that the RSTACT[1:0] field has been updated on the reception of a broadcast or direct write RSTACT CCC (target reset action) and is valid.

This bit is cleared by hardware when the target receives a frame start.

When the device is not in Stop mode:

– If RSTVAL = 1: when RSTF in the I3C_EVR register is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.

– If RSTVAL = 0: when RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.

When in Stop mode, the corresponding interrupt can be used to wake up the device.

Bits 23:22 **RSTACT[1:0]**: Reset action/level on received reset pattern (when the I3C acts as target)

This read field is used by hardware on the reception of a direct read RSTACT CCC in order to return the corresponding data byte on the I3C bus.

This read field is updated by hardware on the reception of a broadcast or direct write RSTACT CCC (target reset action).

Only the defining bytes 0x00, 0x01 and 0x02 are mapped, and RSTACT[1:0] = Defining Byte[1:0].

00: no reset action

01: first level of reset: the application software must either:

a) partially reset the I3C peripheral, by a write and clear of the enable bit of the I3C configuration register (write EN = 0). This resets the I3C bus interface and the I3C kernel sub-parts, without modifying the content of the I3C APB registers (except the EN bit).

b) fully reset the I3C peripheral, including all its registers, via a write and set of the I3C reset control bit of the RCC (reset and clock controller) register.

10: second level of reset: the application software must issue a warm reset, also known as a system reset:

– the software writes and sets the SYSRESETREQ control bit of the AIRCR register, when the device is controlled by a Cortex®-M.

11: no reset action

Bits 21:20 **AS[1:0]**: Activity state (when the I3C acts as target)

This read field is updated by hardware on the reception of a ENTASx CCC (enter activity state, with x = 0-3):

00: activity state 0

01: activity state 1

10: activity state 2

11: activity state 3

Bit 19 **HJEN**: Hot-join request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISHJ= 1 (cleared) and the reception of ENEC CCC with ENHJ= 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: hot-join request disabled

1: hot-join request enabled

Bit 18 Reserved, must be kept at reset value.

Bit 17 **CREN**: Controller-role request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISCR = 1 (cleared) and the reception of ENEC CCC with ENCR = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: controller-role request disabled

1: controller-role request enabled

Bit 16 **IBIEN**: IBI request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISINT = 1 (cleared) and the reception of ENEC CCC with ENINT = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: IBI request disabled

1: IBI request enabled

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 **DA[6:0]**: 7-bit dynamic address

When the I3C acts as controller, this field can be written by software, for defining its own dynamic address.

When the I3C acts as target, this field is updated by hardware on the reception of either the broadcast ENTDAA CCC or the direct SETNEWDA CCC.

Bit 0 **DAVAL**: Dynamic address is valid (when the I3C acts as target)

When the I3C acts as controller, this bit can be written by software, for validating its own dynamic address, for example before a controller-role hand-off.

When the I3C acts as target, this bit is asserted by hardware on the acknowledge of the broadcast ENTDAA CCC or the direct SETNEWDA CCC, and this field is cleared by hardware on the acknowledge of the broadcast RSTDAA CCC.

45.16.18 I3C device x characteristics register (I3C_DEVRx)

Address offset: 0x060 + 0x4 * x, (x = 1 to 4)

Reset value: 0x0000 0000

When the I3C peripheral acts as controller, this register is used to define and store some characteristics of a device target x with their related management from the controller, to communicate accordingly with any of this target x over the I3C bus. Then, the hardware can autonomously identify and acknowledge an allowed IBI or/and controller-role request from a target x, receive the expected IBI payload data if any, and notify the software via the corresponding flag IBIF/CRF (and the corresponding interrupt if enabled) in the I3C_EVR register.

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

Bit 31 **DIS**: Disables writes to DA[6:0] (when the I3C acts as controller)

When the I3C acts as controller, once the software sets IBIACK= 1 or CRACK= 1, this read bit is set by hardware (DIS= 1) to lock the configured DA[6:0] and IBIDEN values.

Then, to be able to modify DA[6:0], IBIDEN or SUSP, the software must wait for DIS to be de-asserted by hardware (polling on DIS= 0) before modifying these three assigned values to the target x. Indeed, the target can request an IBI or a controller-role while the controller intends to modify DA[6:0], IBIDEN or SUSP.

0: write to DA[7:0] and to IBIDEN in the I3C_DEVRx register is allowed

1: write to DA[7:0] and to IBIDEN is disabled/locked

Bits 30:20 Reserved, must be kept at reset value.

Bit 19 **SUSP**: Suspend/stop I3C transfer on received IBI (when the I3C acts as controller)

When the I3C acts as controller, this bit can be used to receive an IBI from target x with pending read notification feature (received MDB[7:5] = 3'b101).

If this bit is set, when an IBI is received and completed (IBIF = 1 in the I3C_EVR register), a stop is emitted on the I3C bus and both C-FIFO and TX-FIFO are automatically flushed by hardware. The controller execution flow is stopped, even if a next control message is programmed. When the IBI is completed, the controller software can issue a new control word, such as a private read, to the target device that initiated the IBI request.

0: C-FIFO and TX-FIFO are not flushed after an IBI request from target x is acknowledged and completed, and depending on the presence or absence of a next control word, a repeated start or a stop is emitted

1: I3C transfer is stopped and both C-FIFO and TX-FIFO are flushed after receiving an IBI request from target x

Bit 18 **IBIDEN**: IBI data enable (when the I3C acts as controller)

When the I3C acts as controller, this bit must be written by software to store the BCR[2] bit as received from the target x during broadcast ENTDAA or direct GETBCR CCC via the received I3C_RDR.

Writing to this field has no impact when the DIS = 1 in the I3C_DEVRx register.

0: no data byte follows the acknowledged IBI from target x

1: the mandatory data byte MDB[7:0] follows the acknowledged IBI from target x

Bit 17 **CRACK**: Controller-role request acknowledge (when the I3C acts as controller)

When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of a controller-role request from target x:

0: a controller-role request from target x must be NACK-ed

After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another controller-role request later on)

1: a controller-role request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed

- The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.

- After the ACK, the message continues as initially programmed. The software is notified by the controller-role request flag (CRF = 1 in the I3C_EVR register) and/or the corresponding interrupt if enabled; For effectively granting the controller-role to the requesting secondary controller, software must issue a GETACCCR (formerly known as GETACCMST), followed by a stop.

- Independently of CRACK configuration for this or other devices, further controller-role request(s) are NACK-ed until controller-role request flag (CRF) and IBI flag (IBIF) in the I3C_EVR register are both cleared.

Bit 16 **IBIACK**: IBI request acknowledge (when the I3C acts as controller)

When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of an IBI request from target x:

0: an IBI request from target x must be NACK-ed

- After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another IBI request later on)

1: an IBI request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed

- The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.
- After the ACK, the controller logs the IBI payload data, if any, depending on I3C_DEVRx.IBIDEN.
- The software is notified by the IBI flag (IBIF = 1) and/or the corresponding interrupt if enabled;
- Independently from IBIACK configuration for this or other devices, further IBI request(s) are NACK-ed until IBI request flag (IBIF) and controller-role request flag (CRF) are both cleared.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 **DA[6:0]**: Assigned I3C dynamic address to target x (when the I3C acts as controller)

When the I3C acts as controller, this field must be written by software to store the 7-bit dynamic address that the controller sends via a broadcast ENTDAA or a direct SETNEWDA CCC acknowledged by the target x.

Writing to this field has no impact when the read field DIS = 1 in the I3C_DEVRx register.

Bit 0 Reserved, must be kept at reset value.

45.16.19 I3C maximum read length register (I3C_MAXRLR)

Address offset: 0x090

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum read length value exchanged with the controller during respectively GETMRL or SETMRL CCC. This register is also used to set the IBI data payload size.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

When receiving a private read message, the target ends the data transmission (by driving T-bit = 0) when the count of transmitted data reaches MRL[15:0] in the I3C_MAXRLR register.

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **IBIP[2:0]**: IBI payload data maximum size, in bytes (when I3C acts as target)
This field is initially written by software when EN = 0 to set the maximum number of data bytes to be sent to the controller after an IBI request has been acknowledged.This field can be updated by hardware on the reception of SETMRL command (which potentially also updated IBIP[2:0]).
Software is notified of an MRL update by MRLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.
000: null payload data size (only allowed when BCR2 = 0 in the I3C_BCR register)
001: 1 byte (mandatory data byte MDB[7:0]
010: 2 bytes (including first MDB[7:0])
011: 3 bytes (including first MDB[7:0])
100: 4 bytes (including first MDB[7:0])
others: same as 100

Bits 15:0 **MRL[15:0]**: Maximum data read length (when I3C acts as target)
This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMRL command (with potentially also updated IBIP[2:0]).
Software is notified of an MRL update by MRLUPF and the corresponding interrupt, if enabled.
This field is used by hardware to return the value on the I3C bus when the target receives a GETMRL CCC.

45.16.20 I3C maximum write length register (I3C_MAXWLR)

Address offset: 0x094

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum write length value exchanged with the controller during respectively GETMWL or SETMWL CCC.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

On receiving a private write message, the target stops the data reception (extra received data are not written into RX-FIFO) when the count of received data reaches MWL[15:0] in the I3C_MAXWLR register.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MWL[15:0]**: Maximum data write length (when I3C acts as target)
This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMWL command.
Software is notified of an MWL update by MWLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.
This field is used by hardware to return the value on the I3C bus when the target receives a GETMWL CCC.

45.16.21 I3C I²C target configuration register (I3C_I2CCFGR)

Address offset: 0x098

Reset value: 0x0000 0000

This register is used to initialize the I3C peripheral with a static I²C address to operate on an I3C or I²C bus. This register can be written only when the I3C peripheral is disabled (i.e. when I3C_CFGR.EN=0).

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

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

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FMP**: high drive enable for Fast-mode plus (when the I3C peripheral is initialized as target)

When the I3C peripheral is initialized as target (I3C_CFGR.CRINIT = 0) and connected on an I²C bus in Fast-mode plus (Fm+), the software can set this bit when the I3C peripheral is disabled (i.e. when I3C_CFGR.EN = 0) to enable the higher 20 mA drive current on SCL and SDA lines.

Note: 20 mA high drive is automatically disabled by the hardware as soon an I3C header (7'hE + RnW = 0) is detected, and this initialization bit is not updated.

0: initialized with a disabled 20 mA drive

1: initialized with an enabled 20 mA drive

Bit 28 **ANFON**: analog filter enable (when the I3C peripheral is initialized as target)

When the I3C peripheral is initialized as target (I3C_CFGR.CRINIT = 0) and connected on an I²C bus at Fm or Fm+ speed, the software can set this bit when the I3C peripheral is disabled (i.e. when I3C_CFGR.EN = 0) to enable the 50 ns spike filter on SCL and SDA lines.

Note: 50 ns spike filter is automatically disabled by the hardware as soon an I3C header (7'hE + RnW = 0) is detected, and this initialization bit is not updated.

0: initialized with a disabled 50 ns spike filter

1: initialized with an enabled 50 ns spike filter

Bits 27:9 Reserved, must be kept at reset value.

Bit 8 **MAVAL**: Minimum I3C target address 7'h01 is valid (when the I3C peripheral is initialized as target)

In case of a single point-to-point I3C bus, when the I3C peripheral is initialized as target (I3C_CFGR.CRINIT = 0) the software can set this bit when the I3C peripheral is disabled (i.e. when I3C_CFGR.EN = 0) to next acknowledge a very basic direct SETDASA with 7'h01 address value.

Note: This initialization bit is not updated by the hardware.

0: Minimum address 7'h01 is not valid: a direct SETDASA with 7'h01 address is not acknowledged

1: Minimum address 7'h01 is valid: a direct SETDASA CCC with 7'h01 address is acknowledged

Bits 7:1 **SA[6:0]**: 7-bit static address (when the I3C peripheral is initialized as target)
When the I3C peripheral is initialized as target (I3C_CFGR.CRINIT = 0) the software can configure this address and writes SAVAL = 1 to set as valid this programmed value, when the I3C peripheral is disabled (i.e. when I3C_CFGR.EN = 0).
This must be written when target is connected on an I^2C bus.
On an I3C bus with a typical ENTDAA dynamic address assignment, this initialized value is not of interest and SAVAL bit must be cleared. On some I3C use cases, an initialized static address may be programmed with SAVAL = 1 before a direct SETDASA or a broadcast SETAASA.

Bit 0 **SAVAL**: Static address is valid (when the I3C peripheral is initialized as target)
When the I3C peripheral is initialized as target (I3C_CFGR.CRINIT = 0) and connected on an I^2C bus, the software must set this bit when the I3C peripheral is disabled (i.e. when I3C_CFGR.EN = 0) to validate the 7-bit static address programmed value.
When the I3C peripheral is initialized as target (I3C_CFGR.CRINIT = 0) and connected on an I3C bus, the software may set this bit when the I3C peripheral is disabled (i.e. when I3C_CFGR.EN = 0) to validate the 7-bit static address programmed value. After I3C detection, the I3C peripheral hardware can be assigned a dynamic value based on the valid programmed SA[6:0] value via the broadcast SETAASA (set all static addresses as the dynamic ones) or via the direct SETDASA (set static address as the dynamic one), as an alternative of the typical ENTDAA usage.
Note: This initialization bit is not updated by the hardware after the I3C dynamic address assignment.
0: initialized without a static address. Typical I3C target usage with ENTDAA
1: initialized with a valid static address on an I^2C bus or an I3C bus with SETDASA/SETAASA

45.16.22 I3C timing register 0 (I3C_TIMINGR0)

Address offset: 0x0A0

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure the SCL clock signal waveform.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCLH_I2C[7:0]								SCLL_OD[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH_I3C[7:0]								SCLL_PP[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **SCLH_I2C[7:0]**: SCL high duration, used for legacy I^2C messages, in number of kernel clocks cycles:
$t_{SCLH_I2C} = (SCLH_I2C + 1) \times t_{I3CCLK}$
Note: SCLH_I2C is used to generate t_{DIG_H} (I^2C) timing when communicating with I^2C devices.
Note: With I^2C Fm+ device t_{DIG_Hmin} = 260 ns, with I^2C Fm+ device t_{DIG_Hmin} = 600 ns.

Bits 23:16 **SCLL_OD[7:0]**: SCL low duration in open-drain phases, used for legacy I^2C messages and for I3C open-drain phases (address phase following a start, ACK phase during controller-initiated messages, and T bit phase during direct/private/IBI payload), in number of kernel clocks cycles:

t_{SCLL_OD} = (SCLL_OD + 1) x t_{I3CCLK}

Note: SCLL_OD is used to generate both t_{DIG_L} (I^2C) and $t_{DIG_OD_L}$(I3C) timings.

Note: With I^2C Fm+ device t_{DIG_Lmin} = 500 ns, with I^2C Fm device t_{DIG_Lmin} = 1320 ns.

Note: I3C messages: $t_{DIG_OD_Lmin}$ = 200 ns.

Note: If a single I3C frame is gathering I^2C and I3C messages, the SCL low duration during I3C open-drain phases is increased to fit I^2C timings.

Bits 15:8 **SCLH_I3C[7:0]**: SCL high duration, used for I3C messages (both in push-pull and open-drain phases), in number of kernel clocks cycles:

t_{SCLH_I3C} = (SCLH_I3C + 1) x t_{I3CCLK}

Note: SCLH_I3C is used to generate both t_{DIG_H} (I3C) and $t_{DIG_H_MIXED}$ timings.

Note: For mixed bus (with at least one I^2C target): $t_{DIG_H_MIXEDmin}$ = 32 ns and $t_{DIG_H_MIXEDmax}$ = 45 ns (due to I^2C 50 ns spike filter).

Note: For pure I3C bus (with no I^2C targets): t_{DIG_Hmin} = 32 ns.

Bits 7:0 **SCLL_PP[7:0]**: SCL low duration in I3C push-pull phases, in number of kernel clocks cycles:

t_{SCLL_PP} = (SCLL_PP + 1) x t_{I3CCLK}

Note: SCLL_PP is used to generate t_{DIG_L} (I3C in PP) timing.

Note: t_{DIG_Lmin} = 32 ns (max 40/60 duty cycle at 12.5 MHz).

45.16.23 I3C timing register 1 (I3C_TIMINGR1)

Address offset: 0x0A4

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure some I3C timing settings.

When the I3C acts as target and is controller-capable, this register is used to configure a timing for the controller-role hand-off procedure.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SDA_HD[1:0]		Res.	Res.	Res.	Res.	Res.	FREE[6:0]						
		rw	rw						rw	rw	rw	rw	rw	rw	rw

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

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **SDA_HD[1:0]**: SDA hold time (when the I3C acts as controller), in number of kernel clocks cycles (refer to MIPI timing SDA hold time in push-pull t_{HD_PP}):

SDA hold time = (SDA_HD[1:0] + 0.5) x t_{I3CCLK}

Note: when controller: t_{HD_PPmiN} = min (t_{CR} , t_{CF}) + 3 ns.

Bits 27:23 Reserved, must be kept at reset value.

Bits 22:16 **FREE[6:0]**: Number of kernel clocks cycles that is used to set some MIPI timings like bus free condition time (when the I3C acts as controller)

When the I3C acts as controller:

1. for I3C start timing: it must wait for (bus free condition) time to be elapsed after a stop and before a start, refer to MIPI timings (I3C) t_{CAS} and (I^2C) t_{BUF}. These timings are defined by: $t_{BUF} = t_{CAS} = [(FREE[6:0] + 1) \times 2 - (0.5 + SDA_HD[1:0])] \times t_{I3CCLK}$

Note: For pure I3C bus: t_{CASmin} = 38.4 ns, and t_{CASmax} = 1 µs, 100 µs, 2 ms, 50 ms for, respectively, ENTAS0, 1, 2, and 3.

Note: For mixed bus with I^2C Fm+ device t_{BUFmin} = 0.5 µs, for mixed bus with I^2C Fm device t_{BUFmin} = 1.3 µs.

2. for I3C repeated start timing: must wait for time to be elapsed after a repeated start (SDA is de-asserted) and before driving SCL low, refer to. MIPI timing t_{CASr}. This timing is defined by: $t_{CASr} = [(FREE[6:0] + 1) \times 2 - (0.5 + SDA_HD[1:0])] \times t_{I3CCLK}$.

Note: $t_{CASr,\ miN}$ = 19.2 ns.

3. for I3C stop timing: must wait for time to be elapsed after that the SCL clock is driven high, and before the stop condition (SDA is asserted). This timing is defined by: $t_{CBP} = (FREE[6:0] + 1) \times t_{I3CCLK}$.

Note: t_{CBPmiN} = 19.2 ns.

4. for I3C repeated start timing (T-bit when controller ends read with repeated start followed by stop): must wait for time to be elapsed after that the SCL clock is driven high, and before the repeated start condition (SDA is de-asserted). This timing is defined by: $t_{CBSr} = (FREE[6:0] + 1) \times t_{I3CCLK}$.

Note: $t_{CBSr,\ miN}$ = 19.2 ns.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **ASNCR[1:0]**: Activity state of the new controller (when I3C acts as active controller)

This field indicates the time to wait before being accessed as new target, refer AVAL[7:0]. This field can be modified only when the I3C acts as controller.

Bits 7:0 **AVAL[7:0]**: Number of kernel clock cycles to set a time unit of 1 µs, whatever I3C acts as controller or target.

This time unit is then used by the hardware to build some internal timers, corresponding to the following MIPI I3C timings:

When the I3C acts as target:

1. for bus available condition time: it must wait for (bus available condition) time to be elapsed after a stop and before issuing a start request for an IBI or a controller-role request (bus free condition is sustained for at least t_{AVAL}). Refer to MIPI timing t_{AVAL} = 1 µs. This timing is defined by: t_{AVAL} = (AVAL[7:0] + 2) x t_{I3CCLK}
2. for bus idle condition time: it must wait for (bus idle condition) time to be elapsed after that both SDA and SCL are continuously high and stable before issuing a hot-join event. Refer to MIPI v1.1 timing t_{IDLE} = 200 µs . This timing is defined by: t_{IDLE} = (AVAL[7:0] + 2) x 200 x t_{I3CCLK}

When the I3C acts as controller, it cannot stall the clock beyond a maximum stall time (stall the SCL clock low), as follows:

1. on first bit of assigned address during dynamic address assignment: it cannot stall the clock beyond the MIPI timing $t_{STALLDAA}$ = 15 ms. This timing is defined by: $t_{STALLDAAmax}$ = (AVAL[7:0] + 1) x 15000 x t_{I3CCLK}
2. on ACK/NACK phase of I3C/I^2C transfer, on parity bit of write data transfer, on transition bit of I3C read transfer: it cannot stall the clock beyond the MIPI timing t_{STALL} = 100 µs. This timing is defined by: $t_{STALLmax}$ = (AVAL[7:0] + 1) x 100 x t_{I3CCLK}

Whatever the I3C acts as controller or as (controller-capable) target, during a controller-role hand-off procedure:

1. The new controller must wait for $t_{NEWCRLock}$ before pulling SDA low (issuing a start) after a completed GETACCR CCC. Then the new controller, within t_{CAS}, can pull SCL low to activate SCL clock. The active controller must wait for the same $t_{NEWCRLock}$ time, or at least 100 µs, before testing if the new controller has gained control of the bus by pulling SDA low. The time to wait depends upon the value of ANSCR[1:0] in the I3C_TIMINGR1 register:
   - ASNCR[1:0] = 00: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x t_{I3CCLK}
   - ASNCR[1:0] = 01: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 100 x t_{I3CCLK}
   - ASNCR[1:0] = 10: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 2000 x t_{I3CCLK}
   - ASNCR[1:0] = 11: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 50000 x t_{I3CCLK}

45.16.24 I3C timing register 2 (I3C_TIMINGR2)

Address offset: 0x0A8

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure and enable SCL clock stalling, to enable SCL clock low stalling, if needed by the addressed I3C or legacy I^2C target(s).

This register can be written only when the I3C acts as controller.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **STALL[7:0]**: Controller clock stall time, in number of kernel clock cycles

t_{SCLL_STALL} = STALL x t_{I3CCLK}

Bit 7 Reserved, must be kept at reset value.

Bit 6 **STALLL:** Controller clock stall enable in the address ACK/NACK phase (before the ninth bit) of a legacy I2C read/write message. This allows the I2C target to prepare the next data byte to send/receive.

0: no stall

1: stall enabled

Bit 5 **STALLS:** Controller clock stall enable in the data ACK/NACK phase of a legacy I2C write message. This allows the I2C target to prepare the next data byte to receive.

0: no stall

1: stall enabled

Bit 4 **STALLR:** Controller clock stall enable in the data ACK/NACK phase of a legacy I2C read message. This allows the I2C target to prepare the next data byte to send.

0: no stall

1: stall enabled

Bit 3 **STALLA:** Controller clock stall enable on ACK phase

The SCL is stalled (during t_{SCLL_STALL} as defined by STALL) in the address ACK/NACK phase (before the ninth bit). This allows the target to prepare data. The ACK driven by the controller itself on a target-initiated request (IBI/HJ/CR) is not impacted by this control bit.

0: no stall

1: stall enabled

Bit 2 **STALLC:** Controller clock stall enable on PAR phase of CCC

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase of common command code (before the ninth bit). This allows the target to decode the command.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

Bit 1 **STALLD:** Controller clock stall enable on PAR phase of Data

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase (before the ninth bit). This allows the target to read received data.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

Bit 0 **STALLT:** Controller clock stall enable on T-bit phase of data

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase (before the ninth bit). This allows the target to prepare the data to be sent.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

45.16.25 I3C bus characteristics register (I3C_BCR)

Address offset: 0x0C0

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure three bits used by hardware to return the data byte BCR[7:0] on reception of GETBCR or ENTDAA CCC. The returned byte BCR[7:0] on the I3C bus is then as follows:

- BCR[7] = 0 (reserved)
- BCR[6] = I3C_BCR6 (controller capable)
- BCR[5] = 1 (advanced capabilities, use GETCAPS CCC to determine which ones)
- BCR[4] = 0 (not a virtual target)
- BCR[3] = 1 (offline capable)
- BCR[2] = I3C_BCR2 (IBI payload)
- BCR[1] = 1 (IBI request capable)
- BCR[0] = I3C_BCR0 (max data speed limitation)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCR6	Res.	Res.	Res.	BCR2	Res.	BCR0
									rw				rw		rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **BCR6**: Controller capable
0: I3C target (no controller capable)
1: I3C controller capable

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **BCR2**: in-band interrupt (IBI) payload
0: no data byte follows the accepted IBI
1: at least one mandatory data byte follows the accepted IBI (and at most 4 data bytes)

Bit 1 Reserved, must be kept at reset value.

Bit 0 **BCR0**: max data speed limitation
0: no limitation
1: limitation, as described by I3C_GETMXDSR.

45.16.26 I3C device characteristics register (I3C_DCR)

Address offset: 0x0C4

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure the device characteristics ID, which is used by hardware to return the data byte DCR[7:0] on reception of GETDCR, ENTDAA, or DEFTGTS CCC.

This register can be written by software only when EN = 0 in the I3C_CFGR register.

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

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

Bits 7:0 **DCR[7:0]**: device characteristics ID
0x00: generic device (for v1.0 devices)
others: ID to describe the type of the I3C sensor/device

Note: The latest MIPI DCR ID assignments are available on https://www.mipi.org.

45.16.27 I3C get capability register (I3C_GETCAPR)

Address offset: 0x0C8

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set the IBI MDB support for pending read notification support, and is used by hardware to return the GETCAP3 byte on reception of GETCAPS CCC, format 1.

The returned byte GETCAP1[7:0] on the I3C bus is then as follows:

- GETCAP1[7:0] = 0 (no HDR)

The returned byte GETCAP2[7:0] on the I3C bus is then as follows:

- GETCAP2[7:6] = 00 (no HDR)
- GETCAP2[5:4] = 00 (no group addressing)
- GETCAP2[3:0] = 0001 (compliant with MIPI specification v1.1)

The returned byte GETCAP3[7:0] on the I3C bus is then as follows:

- GETCAP3[7] = 0 (reserved)
- GETCAP3[6] = CAPPEND in the I3C_GETCAPR register (IBI MDB support for pending read notification)
- GETCAP3[5] = 0 (no HDR)
- GETCAP3[4] = 1 (defining byte support in GETSTATUS)
- GETCAP3[3] = 1 (defining byte support in GETCAPS)
- GETCAP3[2] = 0 (no device-to-device transfer)
- GETCAP3[1] = 0 (no device-to-device transfer)
- GETCAP3[0] = 0 (no multi-lane data transfer)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

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

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **CAPPEND**: IBI MDB support for pending read notification

This bit is written by software during bus initialization (EN = 0), and indicates the support (or not) of the pending read notification via the IBI MDB[7:0] value.

This bit is used to return the GETCAP3 byte in response to the GETCAPS CCC format 1.

0: this I3C when acting as target sends an IBI request without a mandatory data byte value indicating a pending read notification

1: this I3C when acting as target sends an IBI request with a mandatory data byte value (MDB[7:5] = 101), indicating a pending read notification

Bits 13:0 Reserved, must be kept at reset value.

45.16.28 I3C controller-role capability register (I3C_CRCAPR)

Address offset: 0x0CC

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set features the I3C supports as a secondary controller after controller-role hand-off, and is used by hardware to return the CRCAP1 byte and the CRCAP2 byte on reception of GETCAPS CCC, format 2 with the defining byte CRCAPS (0x91).

The returned CRCAP1[7:0] on the I3C bus is then as follows:

- CRCAP1[7:3] = 00000 (reserved)
- CRCAP1[2] = 0 (no multi-lane)
- CRCAP1[1] = CAPGRP in the I3C_CRCAPR register (group management)
- CRCAP1[0] = 1 (hot-join)

The returned CRLCAP2[7:0] on the I3C bus is then as follows:

- CRCAP2[7:4] = 0000 (reserved)
- CRCAP2[3] = CAPDHOFF in the I3C_CRCAPR register (delayed controller-role handoff)
- CRCAP2[2] = 1 (deep sleep capable)
- CRCAP2[1] = 0 (no automatic controller-role pass-back)
- CRCAP2[0] = 1 (IBI ack capable)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

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

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

Bit 9 **CAPGRP**: group management support (when acting as controller)

This bit is written by software during bus initialization (EN = 0), and indicates if the I3C is able to support group management when it acts as a controller (after controller-role hand-off) via emitted DEFGRPA, RSTGRPA, and SETGRPA CCC.

This bit is used to return the CRCAP1 byte in response to the GETCAPS CCC format 2.

0: this I3C does not support group address capabilities

1: this I3C supports group address capabilities (when becoming controller)

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **CAPDHOFF**: delayed controller-role hand-off

This bit is written by software during bus initialization (EN = 0), and indicates if this target I3C needs additional time to process a controller-role hand-off requested by the current controller.

This bit is used to return the CRCAP2 byte in response to the GETCAPS CCC format 2.

0: this I3C does not needs additional time to process a controller-role hand-off

1: this I3C needs additional time to process a controller-role hand-off

Bits 2:0 Reserved, must be kept at reset value.

45.16.29 I3C get max data speed register (I3C_GETMXDSR)

Address offset: 0x0D0

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set its capabilities and limitations if any. This register is used by hardware to return data byte(s) on reception of GETMXDS CCC, with format 1 (2 bytes, without maxRdTurn), format 2 (5 bytes, with MaxRdTurn), or format 3 (with defining byte WRRDTURN = 0x00: same 5 bytes as format 2, or with defining byte CRHDLY = 0x91: single byte CRHDLY1).

The returned byte MaxWr[7:0] on the I3C bus is then as follows:

- MaxWr[7:4] = 0000 (reserved)
- MaxWr[3] = 1 (defining byte WRRDTURN and CRHDLY support)
- MaxWr[2:0] = 000 (max sustained data rate for non-CCC messages sent by the controller to the target is designed to operate at 12.5 MHz)

The returned byte MaxRd[7:0] on the I3C bus is then as follows:

- MaxRd[7] = 0 (reserved)
- MaxRd[6] = 1 (stop is allowed between write and read)
- MaxRd[5:3] = 100 if TSCO = 0 (clock to data turnaround time t_{SCO} ≤ 12 ns) in the I3C_GETMXDSR register, else 111 (t_{SCO} > 12 ns, refer to the datasheet for more details)
- MaxRd[2:0] = 000 (max sustained data rate for non-CCC messages sent by the target to the controller is designed to operate at 12.5 MHz)

The returned 3-byte MaxRdTurn[23:0], if FMT[1:0] = 00 in the I3C_GETMXDSR register, on the I3C bus is then as follows:

- MaxRdTurn[23:0], with either the MSB (between 65 ms and 16 s), the middle byte (between 256 µs and 65 ms), or the LSB (less than 256 µs) from RDTURN[7:0] in the I3C_GETMXDSR register (others bits are 0), and depending upon FMT[1:0] in the same register.

The returned byte CRHDLY1[7:0] on the I3C bus is then as follows:

- CRHDLY1[7:3] = 00000 (reserved)
- CRHDLY1[2] = 0 if HOFFAS[1:0] = 00 in the I3C_GETMXDSR register, else 1 (set bus activity state)
- CRHDLY1[1:0] = HOFFAS[1:0] (controller-role hand-off activity state)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

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

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TSCO**: clock-to-data turnaround time (t_{SCO})

This bit is written by software during bus initialization (EN = 0 in the I3C_CFGR register) and is used to specify the clock-to-data turnaround time t_{SCO} (vs. the value of 12 ns). This bit is used by the hardware in response to the GETMXDS CCC to return the encoded clock-to-data turnaround time via the returned MaxRd[5:3] bits.

0: $t_{SCO} \leq 12$ ns

1: $t_{SCO} > 12$ ns (refer to the datasheet for more details)

Bits 23:16 **RDTURN[7:0]**: programmed byte of the 3-byte MaxRdTurn (maximum read turnaround byte)

This bit is written by software during bus initialization (EN = 0) and writes the value of the selected byte (via the FMT[1:0] field) of the 3-byte MaxRdTurn, which is returned in response to the GETMXDS CCC format 2 to encode the maximum read turnaround time.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **FMT[1:0]**: GETMXDS CCC format

This field is written by software during bus initialization (EN = 0) and indicates how is returned the GETMXDS format 1 (without MaxRdTurn) and format 2 (with MaxRdTurn).

This field is used to return the 2-byte format 1 (MaxWr, MaxRd) or 5-byte format 2 (MaxWr, MaxRd, 3-byte MaxRdTurn) byte in response to the GETCAPS CCC.

00: format 1 (2 bytes with MaxWr with no defining byte, MaxRd)

01: format 2: (5 bytes w.ith MaxWr with no defining byte, MaxRd, MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = 0 and LSB = RDTURN[7:0].
- Max read turnaround time is less than 256 µs.

10: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, and middle byte of MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = RDTURN[7:0] and LSB = 0.
- Max read turnaround time is between 256 and 65535 µs

11: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, MSB of MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = RDTURN[7:0], middle byte = 0 and LSB = 0.
- Max read turnaround time is between 65535 µs and 16 s.

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

Bits 1:0 **HOFFAS[1:0]**: Controller hand-off activity state

This field is written by software during bus initialization (EN = 0), and indicates in which initial activity state the (other) current controller must expect the I3C bus after a controller-role hand-off to this controller-capable I3C, when returning the defining byte CRHDLY (0x91) to a GETMXDS CCC.

This 2-bit field is used to return the CRHDLY1 byte in response to the GETCAPS CCC format 3, to state the activity state of this I3C when becoming controller after a controller-role hand-off, and consequently the time the former controller must wait before testing this I3C to be confirmed its ownership.

00: activity state 0 is the initial activity state of this I3C before and when becoming controller

01: activity state 1 is the initial activity state of this I3C when becoming controller

10: activity state 2 is the initial activity state of this I3C when becoming controller

11: activity state 3 is the initial activity state of this I3C when becoming controller

45.16.30 I3C extended provisioned ID register (I3C_EPIDR)

Address offset: 0xD4

Reset value: 0x0208 0000

When the I3C acts as target, this register is used to set the 4-bit MIPI instance ID by software, and some other constant bits used for the 48-bit provisioned ID. It is also used by hardware to return the six bytes for the 48-bit provisioned ID on reception of GETPID and ENTDAA CCC.

The 48-bit provisioned ID on the I3C bus is then returned as:

- provisioned ID [47:32] = 0x0208
- provisioned ID [31:16] = 0x1381
- provisioned ID [15:0] = {MIPIID[3:0], 0x000}

This register can be written by software only when EN = 0 in the I3C_CFGR register.

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

Bits 31:17 **MIPIMID[14:0]**: 15-bit MIPI manufacturer ID

This read field is the 15-bit STMicroelectronics MIPI ID (0x0104).

This field represents bits[47:33] of the 48-bit provisioned ID.

Bit 16 **IDTSEL**: provisioned ID type selector

This field is set as 0 (vendor fixed value).

This field represents bit[32] of the 48-bit provisioned ID.

Note: Bits[31:16] of the provisioned ID can be 0.

Bits 15:12 **MIPIID[3:0]**: 4-bit MIPI Instance ID

This field is written by software to set and identify individually each instance of this I3C IP with a specific number on a single I3C bus.

This field represents bits[15:12] of the 48-bit provisioned ID.

Note: Bits[11:0] of the provisioned ID can be 0.

Bits 11:0 Reserved, must be kept at reset value.

45.16.31 I3C register map

Table 531. I3C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	I3C_CR	MEND	MTYPE[3:0]				Res.	Res.	Res.	ADD[6:0] / CCC[7:0]							RNW	DCNT[15:0]																
	Reset value	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	I3C_CFGR	Res.	TSFSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCFDIS	CFLUSH	CDMAEN	TMODE	SMODE	SFLUSH	SDMAEN	Res.	TXTHRES	TXFLUSH	TXDMAEN	Res.	RXTHRES	RXFLUSH	RXDMAEN	HJACK	Res.	HKSDAEN	EXITPTRN	RSTPTRN	NOARBH	CRINIT	EN
	Reset value		0								0	0	0	0	0	0	0		0	0	0		0	0	0	0		0	0	0	0	0	0
0x010	I3C_RDR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RDB0[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	I3C_RDWR	RDB3[7:0]								RDB2[7:0]								RDB1[7:0]								RDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	I3C_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDB0[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	I3C_TDWR	TDB3[7:0]								TDB2[7:0]								TDB1[7:0]								TDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	I3C_IBIDR	IBIDB3[7:0]								IBIDB2[7:0]								IBIDB1[7:0]								IBIDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	I3C_TGTTDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRELOAD	TGTTDCNT[15:0]															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	I3C_SR	MID[7:0]								Res.	Res.	Res.	Res.	Res.	DIR	ABT	Res.	XDCNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0						0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	I3C_SER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERR	DNACK	ANACK	COVR	DOVR	STALL	PERR	CODERR[3:0]			
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x040	I3C_RMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADD[6:0]							s_e_TR	RCODE[7:0]								Res.	Res.	Res.	Res.	Res.	IBIRDCNT[2:0]		
	Reset value									0	0	0	0	0	0	0		0	0	0	0	0	0	0	0						0	0	0

Table 531. I3C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x050	I3C_EVR	GRPF	DEFF	INTUPDF	ASUPDF	RSTF	MRLUPDF	MWLUPDF	DAUPDF	STAF	GETF	WKPF	Res.	HJF	CRUPDF	CRF	IBIENDF	IBIF	Res.	Res.	Res.	ERRF	RXTGTENDF	FCF	Res.	RXLASTF	TXLASTF	RXFNEF	TXFNFF	SFNEF	CFNFF	TXFEF	CFEF
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0		0	0	0	0	0	0	1	1
0x054	I3C_IER	GRPIE	DEFIE	INTUPDIE	ASUPDIE	RSTIE	MRLUPDIE	MWLUPDIE	DAUPDIE	STAIE	GETIE	WKPIE	Res.	HJIE	CRUPDIE	CRIE	IBIENDIE	IBIIE	Res.	Res.	Res.	ERRIE	RXTGTENDIE	FCIE	Res.	Res.	Res.	RXFNEIE	TXFNFIE	SFNEIE	CFNFIE	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0				0	0	0	0		
0x058	I3C_CEVR	CGRPF	CDEFF	CINTUPDF	CASUPDF	CRSTF	CMRLUPDF	CMWLUPDF	CDAUPDF	CSTAF	CGETF	CWKPF	Res.	CHJF	CRRUPDF	CCRF	CIBIENDF	CIBIF	Res.	Res.	Res.	CERRF	CRXTGTENDF	CFCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0									
0x05C	I3C_MISR	GRPMIS	DEFMIS	INTUPDMIS	ASUPDMIS	RSTMIS	MRLUPDMIS	MWLUPDMIS	DAUPDMIS	STAMIS	GETMIS	WKPMIS	Res.	HJMIS	CRUPDMIS	CRMIS	IBIENDMIS	IBIMIS	Res.	Res.	Res.	ERRMIS	RXTGTENDMIS	FCMIS	Res.	Res.	Res.	RXFNEMIS	TXFNFMIS	SFNEMIS	CFNFMIS	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0				0	0	0				0	0	0	0		
0x060	I3C_DEVR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTVAL	RSTACT [1:0]		AS [1:0]		HJEN	Res.	CREN	IBIEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							DAVAL
	Reset value								0	0	0	0	0	0		0	0									0	0	0	0	0	0	0	0
0x060 + 4 * x, (x = 1 to 4)	I3C_DEVRx	DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	IBIDEN	CRACK	IBIACK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							Res.
	Reset value	0												0	0	0	0									0	0	0	0	0	0	0	
0x090	I3C_MAXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IBIP [2:0]			MRL15:0]															
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	I3C_MAXWLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MWL15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x098	I3C_I2CCFGR	Res.	Res.	FMP	ANFON	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAVAL	SA[6:0]							SAVAL
	Reset value			0	0																				0	0	0	0	0	0	0	0	0
0x0A0	I3C_TIMINGR0	SCLH_I2C[7:0]								SCLL_OD[7:0]								SCLH_I3C[7:0]								SCLL_PP[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	I3C_TIMINGR1	Res.	Res.	SDA_HD		Res.	Res.	Res.	Res.	Res.	FREE[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	ASNCR[1:0]		AVAL[7:0]							
	Reset value			0	0						0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0

Table 531. I3C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0A8	I3C_TIMINGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AVAL[7:0]								Res.	STALL	STALS	STALR	STALA	STALC	STALD	STALT
	Reset value																	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x0C0	I3C_BCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCR6	Res.	Res.	Res.	BCR2	Res.	BCR0
	Reset value																										0				0		0
0x0C4	I3C_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0C8	I3C_GETCAPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPPEND	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0														
0x0CC	I3C_CRCAPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPGR	Res.	Res.	Res.	Res.	Res.	CAPDHOFF	Res.	Res.	Res.
	Reset value																							0						0			
0x0D0	I3C_GETMXDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCO	RDTURN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	FMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	HOFFAS [1:0]	
	Reset value								0	0	0	0	0	0	0	0	0							0	0							0	0
0x0D4	I3C_EPIDR	MIPIMID[14:0]															IDTSEL	MIPIID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0												

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

46 Universal synchronous/asynchronous receiver transmitter (USART/UART)

46.1 USART introduction

The USART offers a flexible means to perform full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and half-duplex single-wire communications, as well as LIN (local interconnection network), smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

46.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data

  Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  – Transmits parity bit

  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode

46.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability (compatible with all LIN specification versions; complete LIN support requires additional software management).
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T = 0 and T = 1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

46.4 USART implementation

The following tables describe the USART implementation. LPUART is included for comparison.

Table 532. Instance implementation on STM32H7Pxxx

Instance	STM32H7Pxxx
USART1	Full
USART2	Full
USART3	Full
USART6	Full
USART10	Full
UART4	Basic
UART5	Basic
UART7	Basic
UART8	Basic
UART9	Basic
LPUART1	Low-power

Table 533. USART/LPUART features

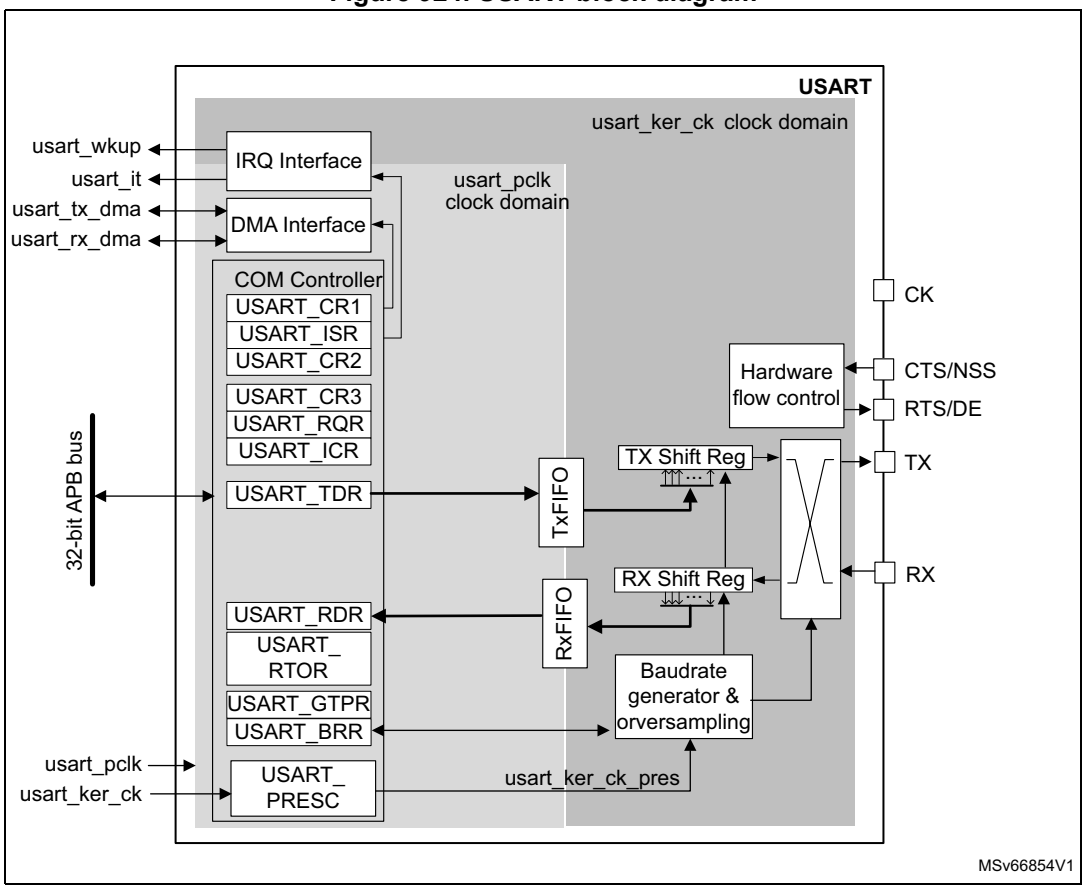
Modes/features[1]	Full feature	Basic feature	Low-power feature
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (master/slave)	X	-	-
Smartcard mode	X	-	-
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual clock domain	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	16		
Wake-up from low-power mode	X[2]	X[2]	X[2]

1. “X” = supported, “-” = not supported.
2. Wake-up supported from Stop mode.

46.5 USART functional description

46.5.1 USART block diagram

Figure 924. USART block diagram



46.5.2 USART pins and internal signals

Description of USART input/output pins

- USART bidirectional communications

  USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

  – **RX** (Receive Data Input)

    RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.

  – **TX** (Transmit Data Output)

    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In single-wire and smartcard modes, this I/O is used to transmit and receive data.

- RS232 hardware flow control mode

  The following pins are required in RS232 hardware flow control mode:

  – **CTS** (Clear To Send)

    When driven high, this signal blocks the data transmission at the end of the current transfer.

  – **RTS** (Request To Send)

    When it is low, this signal indicates that the USART is ready to receive data.

- RS485 hardware control mode

  The **DE** (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

- Synchronous SPI master/slave mode and smartcard mode

  The following pins are required in synchronous master/slave mode and smartcard mode:

  – **CK**

    This pin acts as clock output in synchronous SPI master and smartcard modes.

    It acts as clock input in synchronous SPI slave mode.

    In synchronous master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (such as LCD drivers). The clock phase and polarity are software programmable.

    In smartcard mode, CK output provides the clock to the smartcard.

  – **NSS**

    This pin acts as Slave Select input in synchronous slave mode.

Refer to *Table 534* and *Table 535* for the list of USART input/output pins and internal signals.

Table 534. USART/UART input/output pins

Pin name	Signal type	Description
USART_RX/UART_RX	Input	Serial data receive input
USART_TX/UART_TX	Output	Transmit data output
USART_CTS/UART_CTS	Input	Clear to send
USART_RTS/UART_RTS	Output	Request to send
USART_DE(1)/UART_DE(2)	Output	Driver enable
USART_CK	Output	Clock output in synchronous master and smartcard modes.
USART_NSS(3)	Input	Slave select input in synchronous slave mode.

1. USART_DE and USART_RTS share the same pin.
2. UART_DE and UART_RTS share the same pin.
3. USART_NSS and USART_CTS share the same pin.

Description of USART input/output signals

Table 535. USART internal input/output signals

Signal name	Signal type	Description
usart_pclk	Input	APB clock
usart_ker_ck	Input	USART kernel clock
usart_wkup	Output	USART provides a wake-up interrupt
usart_it	Output	USART global interrupt
usart_tx_dma	Input/output	USART transmit DMA request
usart_rx_dma	Input/output	USART receive DMA request

46.5.3 USART clocks

The simplified block diagram given in *Section 46.5.1: USART block diagram* shows two fully-independent clock domains:

- The **usart_pclk** clock domain

  The **usart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.
- The **usart_ker_ck** kernel clock domain.

  The **usart_ker_ck** is the USART clock source. It is independent from **usart_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart_ker_ck** clock is stopped.

  When the dual clock domain feature is not supported, the **usart_ker_ck** clock is the same as the **usart_pclk** clock.

There is no constraint between **usart_pclk** and **usart_ker_ck**: **usart_ker_ck** can be faster or slower than **usart_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external CK signal provided by the external master SPI device. The **usart_ker_ck** clock must be at least 3 times faster than the clock on the CK input.

46.5.4 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see *Figure 925*):

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

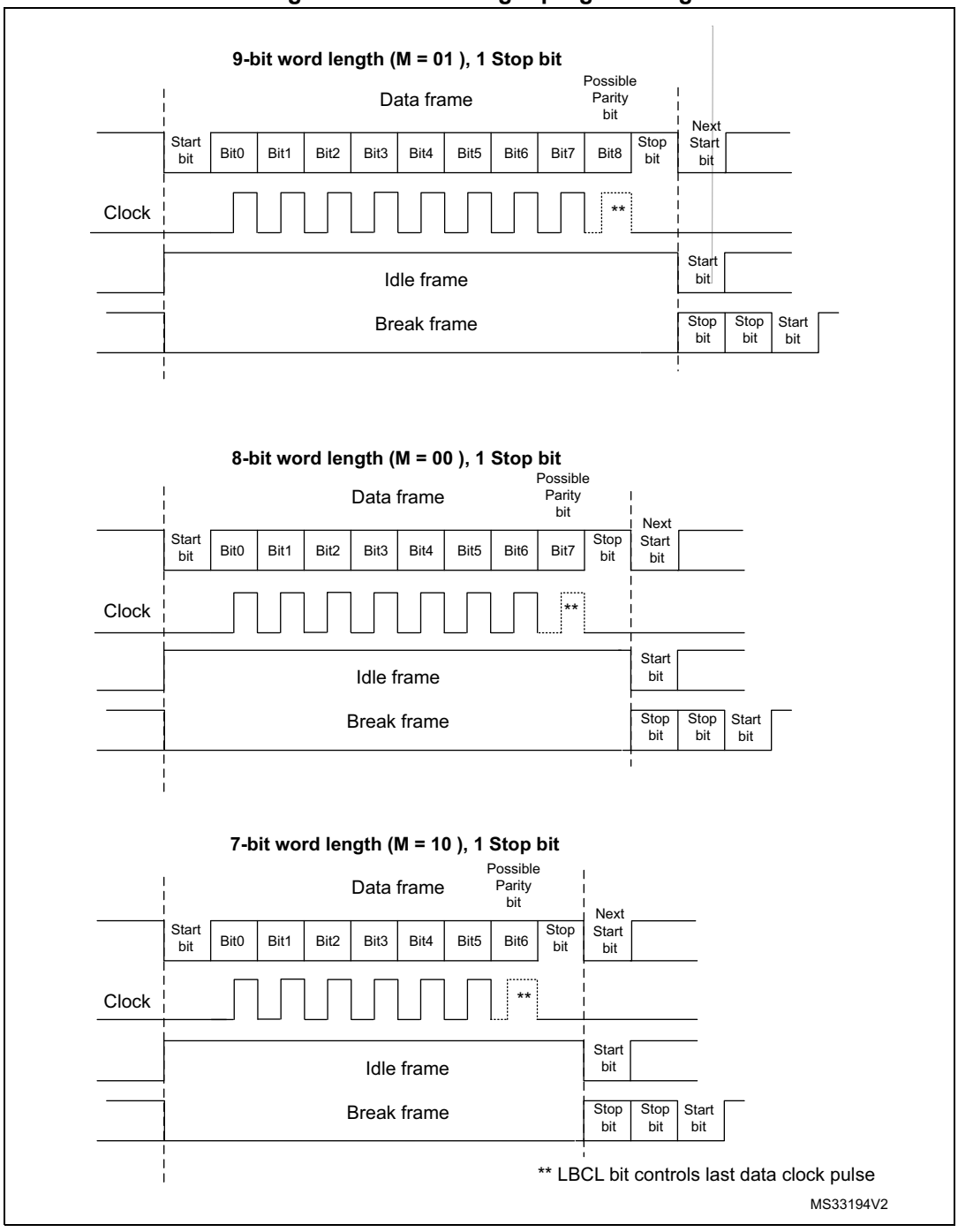
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.

Figure 925. Word length programming



46.5.5 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART_CR1 register (bit 29). This mode is supported only in UART, SPI and smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the USART_ISR register.

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the USART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield.

  In this case, the RXFT flag is set in the USART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in USART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when RXFTCFG[2:0] is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size – 1 data in the RXFIFO and 1 data in the USART_RDR). As a result, the next received data does not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the USART_CR3 register.

46.5.6 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the CK pin.

Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the USART_TDR.

The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost.

An idle frame is sent when the TE bit is enabled.

Configurable stop bits

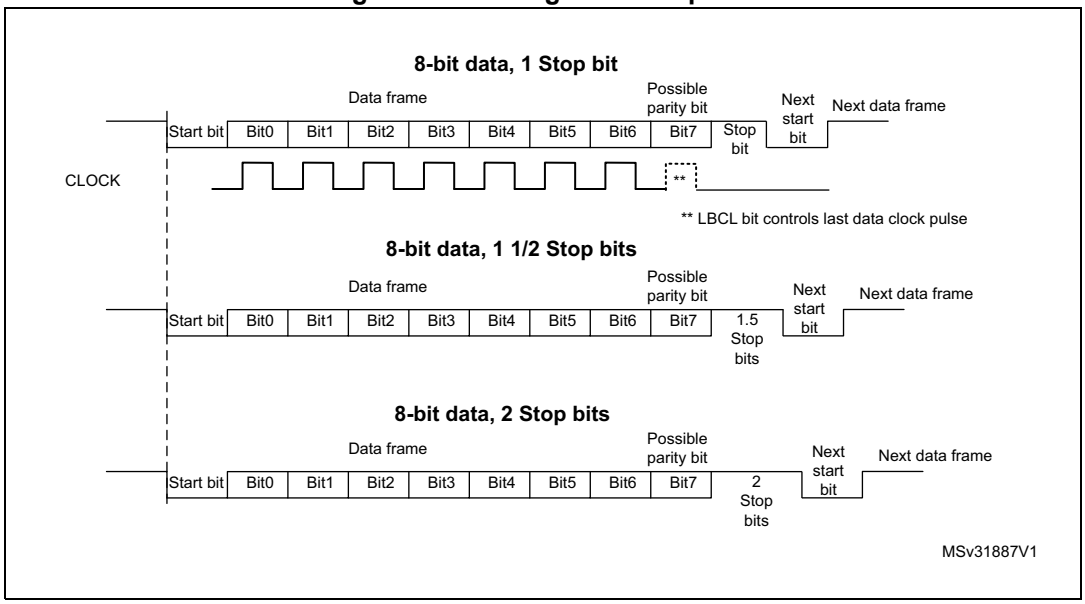
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 stop bits***: This is supported by normal USART, single-wire and modem modes.
- ***1.5 stop bits***: To be used in smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits (see *Section 46.5.1: USART block diagram*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 926. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if multibuffer communication must take place. Configure the DMA register as explained in *Section 46.5.20: Continuous communication using USART and DMA*.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.

7. Write the data to send in the USART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data to the USART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data to the USART_TDR adds one data to the TXFIFO. Write operations to the USART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the USART_TDR register, wait until TC = 1.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame has completed.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode is disabled

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:
  – the data have been moved from the USART_TDR register to the shift register and the data transmission has started;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data.

  This flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the USART_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the USART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.
- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  – the TXFIFO is not full;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

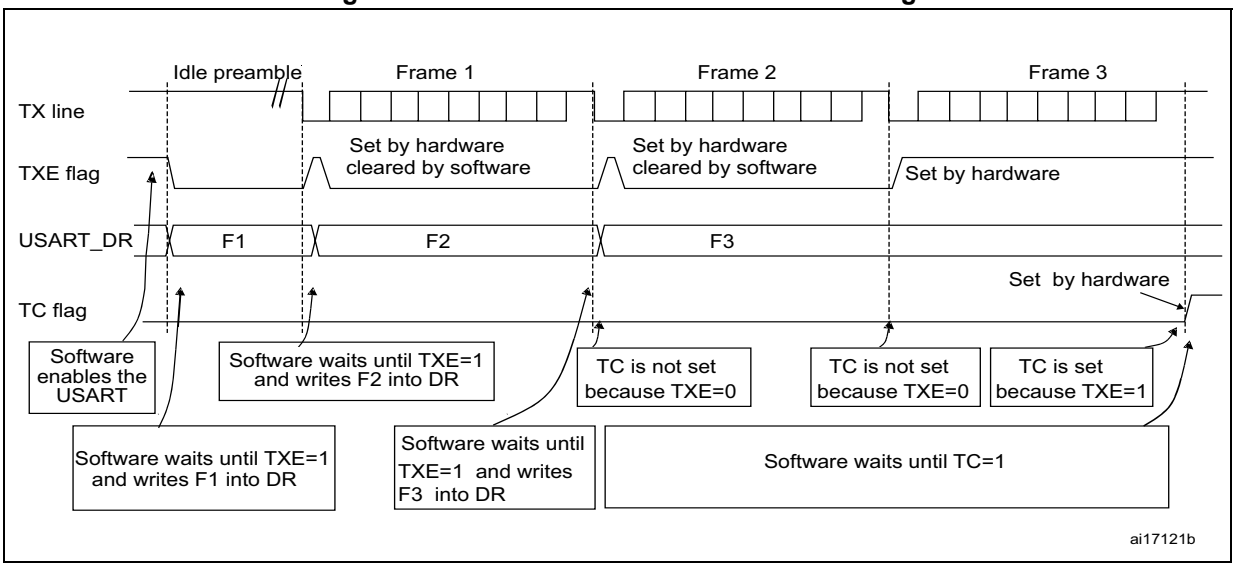
  When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.

  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

After writing the last data to the USART_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the microcontroller to enter the low-power mode (see *Figure 927*).

Figure 927. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see *Figure 925*).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

46.5.7 USART receiver

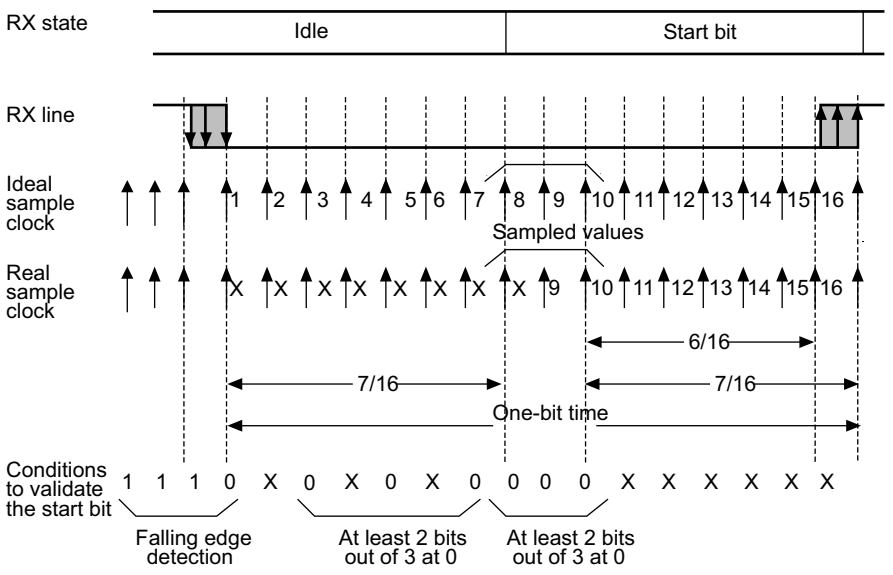
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.

Figure 928. Start bit detection when oversampling by 16 or 8



Note: If the sequence has not completed, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE = 1, or RXFNE flag set and interrupt generated if RXFNEIE = 1 if FIFO mode enabled) if the 3 sampled bits are at 0 (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at 0 and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at 0).

The start bit is validated but the NE noise flag is set if,

- for both samplings, 2 out of the 3 sampled bits are at 0 (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits), or
- for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at 0.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 46.5.20: Continuous communication using USART and DMA*.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty, that is when there are data to be read from the RXFIFO.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to 1 in the USART_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from USART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to 1 in USART_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be

generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs if a character is received and RXNE has not been reset.

  Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte reception.

  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – the ORE bit is set;
  – the RDR content is not lost. The previous data is available by reading the USART_RDR register.
  – the shift register is overwritten. After that, any data received during overrun is lost.
  – an interrupt is generated if either the RXNEIE or the EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.

  Data can not be transferred from the shift register to the USART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set.
  – The first entry in the RXFIFO is not lost. It is available by reading the USART_RDR register.
  – The shift register is overwritten. After that point, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USART_ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost.

When the FIFO mode is disabled, there are two possibilities

- *if RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE = 0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.*

Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see *Section : Reset and Clock Control (RCC)*). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

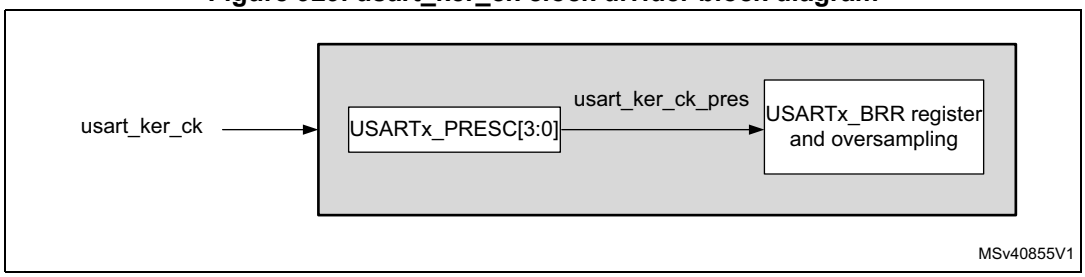
- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is usart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the usart_ker_ck clock source can be configurable in the RCC (see *Section : Reset and Clock Control (RCC)*). Otherwise the usart_ker_ck clock is the same as usart_pclk.

The usart_ker_ck clock can be divided by a programmable factor, defined in the USART_PRESC register.

Figure 929. usart_ker_ck clock divider block diagram



Some usart_ker_ck sources enable the USART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the USART_RDR register or by DMA.

For the other clock sources, the system must be active to enable USART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This enables obtaining the best a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register either to 16 or 8 times the baud rate clock (see *Figure 930* and *Figure 931*).

Depending on the application:

- select oversampling by 8 (OVER8 = 1) to achieve higher speed (up to usart_ker_ck_pres/8). In this case the maximum receiver tolerance to clock deviation is reduced (refer to *Section 46.5.9: Tolerance of the USART receiver to clock deviation*)
- select oversampling by 16 (OVER8 = 0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum usart_ker_ck_pres/16 (where usart_ker_ck_pres is the USART input clock divided by a prescaler).

Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.
- A single sample in the center of the received bit

  Depending on the application:

  – select the three sample majority vote method (ONEBIT = 0) when operating in a noisy environment and reject the data when a noise is detected (refer to *Table 536*) because this indicates that a glitch occurred during the sampling.

  – select the single sample method (ONEBIT = 1) when the line is noise-free to increase the receiver tolerance to clock deviations (see *Section 46.5.9: Tolerance of the USART receiver to clock deviation*). In this case the NE bit is never set.

When noise is detected in a frame:

- The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The NE bit is reset by setting NFCF bit in ICR register.

Note: Noise error is not supported in SPI and IrDA modes.

Oversampling by 8 is not available in the smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to 0 by hardware.

Figure 930. Data sampling when oversampling by 16

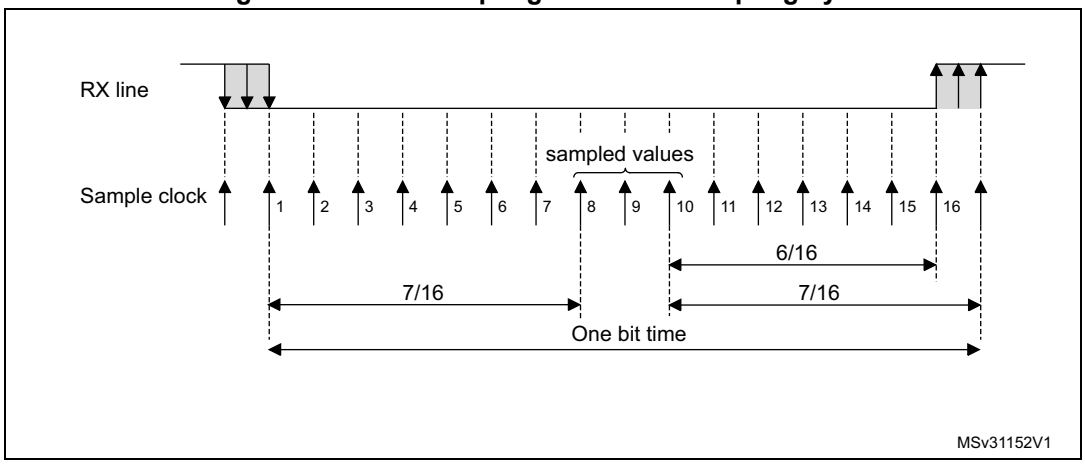


Figure 931. Data sampling when oversampling by 8

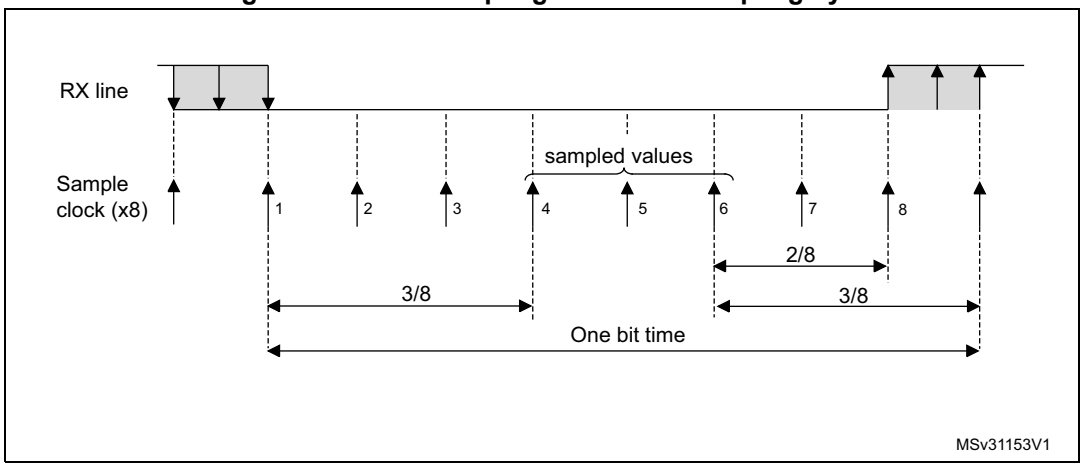


Table 536. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware.
- the invalid data is transferred from the Shift register to the USART_RDR register (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the USART_ICR register.

Note: Framing error is not supported in SPI mode.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of USART_CR: it can be either 1 or 2 in normal mode and 0.5 or 1.5 in smartcard mode.

- **0.5 stop bit (reception in smartcard mode)**: no sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.
- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **1.5 stop bits (smartcard mode)**

  When transmitting in smartcard mode, the device must check that the data are correctly sent. The receiver block must consequently be enabled (RE = 1 in USART_CR1) and the stop bit is checked to test if the smartcard has detected a parity error.

  In the event of a parity error, the smartcard forces the data signal low during the sampling (NACK signal), which is flagged as a framing error. The FE flag is then set through RXNE flag (RXFNE if the FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be broken into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through (refer to *Section 46.5.17: USART receiver timeout* for more details).
- **2 stop bits**

  Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit.

  The framing error flag is set if a framing error is detected during the first stop bit.

  The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

46.5.8 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART_BRR register.

Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = 0 or 1)

In case of oversampling by 16, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

Equation 2: baud rate in smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value must not be changed during communication.

In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16.

How to derive USARTDIV from USART_BRR register values

Example 1

To obtain 9600 bauds with usart_ker_ck_pres= 8 MHz:

- In case of oversampling by 16:
  USARTDIV = 8 000 000/9600
  BRR[15:0] = USARTDIV = 0d833 = 0x0341
- In case of oversampling by 8:
  USARTDIV = 2 * 8 000 000/9600
  USARTDIV = 1666,66 (0d1667 = 0x0683)
  BRR[3:0] = 0x3 >>1 = 0x1
  BRR[15:0] = 0x0681

Example 2

To obtain 921.6 kbauds with usart_ker_ck_pres = 48 MHz:

- In case of oversampling by 16:
  USARTDIV = 48 000 000/921 600
  BRR[15:0] = USARTDIV = 52 = 0x0034
- In case of oversampling by 8:
  USARTDIV = 2 * 48 000 000/921 600
  USARTDIV = 104 (0d104 = 0x68)
  BRR[3:0] = USARTDIV[3:0] >> 1 = 0x8 >> 1 = 0x4
  BRR[15:0] = 0x0064

46.5.9 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < USART\ receiver\ tolerance$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WUUSART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUUSART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUUSART}}{9 \times Tbit}$$

$t_{WUUSART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 537*, *Table 538*, depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.

Table 537. Tolerance of the USART receiver when BRR [3:0] = 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16%	4.86%	2.77%	4.16%

Table 538. Tolerance of the USART receiver when BRR[3:0] is different from 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7%	4.31%	2.22%	3.33%

Note: The data specified in Table 537 and Table 538 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M= 01 or 9- bit times when M = 10).

46.5.10 USART auto baud rate detection

The USART can detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/16.
- When oversampling by 8, the baud rate ranges from usart_ker_ck_pres/32763 and usart_ker_ck_pres/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are the following:

- **Mode 0**: Any character starting with a bit at 1.

  In this case the USART measures the duration of the start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.

  In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.
- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3**: A 0x55 character frame.

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a 0).

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection must be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART_ISR register.

Note: The BRR value might be corrupted if the USART is disabled (UE = 0) during an auto baud rate operation.

46.5.11 USART multiprocessor communication

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the USART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two usart_ker_ck cycles), otherwise mute mode might remain active.

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

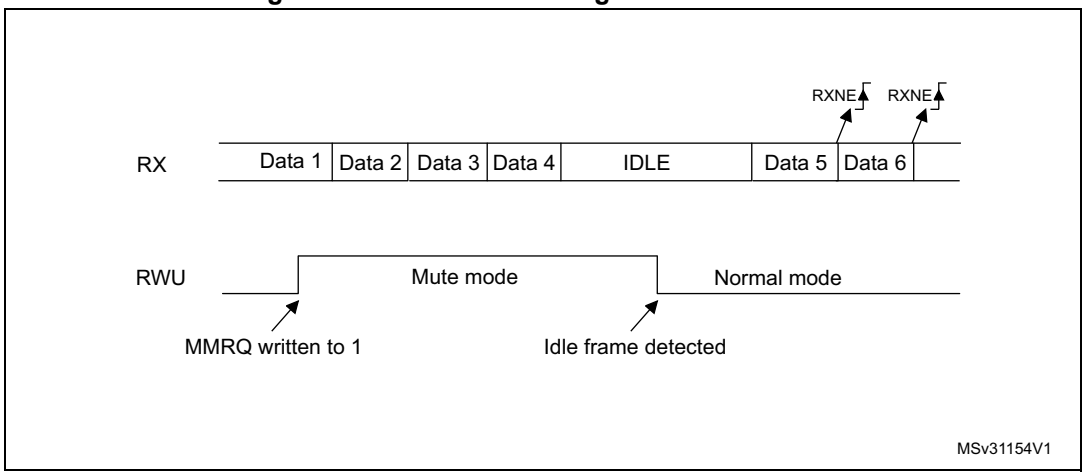
- Idle line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The USART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 932*.

Figure 932. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the USART is activated while the line is idle, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a 1, otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters mute mode. When FIFO management is enabled, the software must ensure that there is at least one empty location in the RXFIFO before entering mute mode.

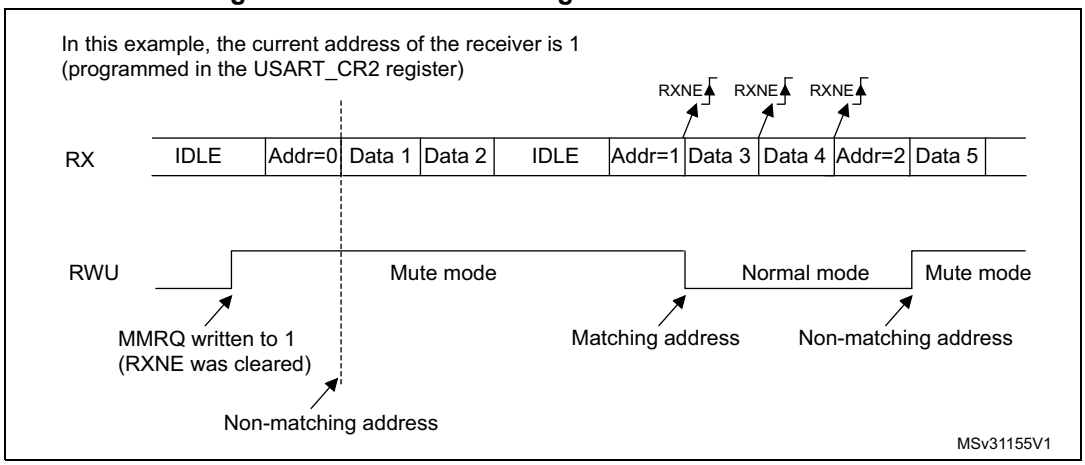
The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in mute mode

An example of mute mode behavior using address mark detection is given in *Figure 933*.

Figure 933. Mute mode using address mark detection



46.5.12 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception has not completed.

Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE = 1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

46.5.13 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in *Table 539*.

Table 539. USART frame formats

M bits	PCE bit	USART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

46.5.14 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 46.4: USART implementation*.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- STOP[1:0] and CLKEN in the USART_CR2 register,
- SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in *Section 46.5.5* has to be applied for LIN master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 zero bits as a break character. Then 2 bits of value ‘1 are sent to enable the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE = 1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL = 1 in USART_CR2) consecutive bits are detected as 0, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit = 1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a 1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN = 0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN = 1), as soon as a framing error occurs (that is, stop bit detected at 0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown in *Figure 934*.

Examples of break frames are given in *Figure 935*.

Figure 934. Break detection in LIN mode (11-bit break length - LBDL bit is set)

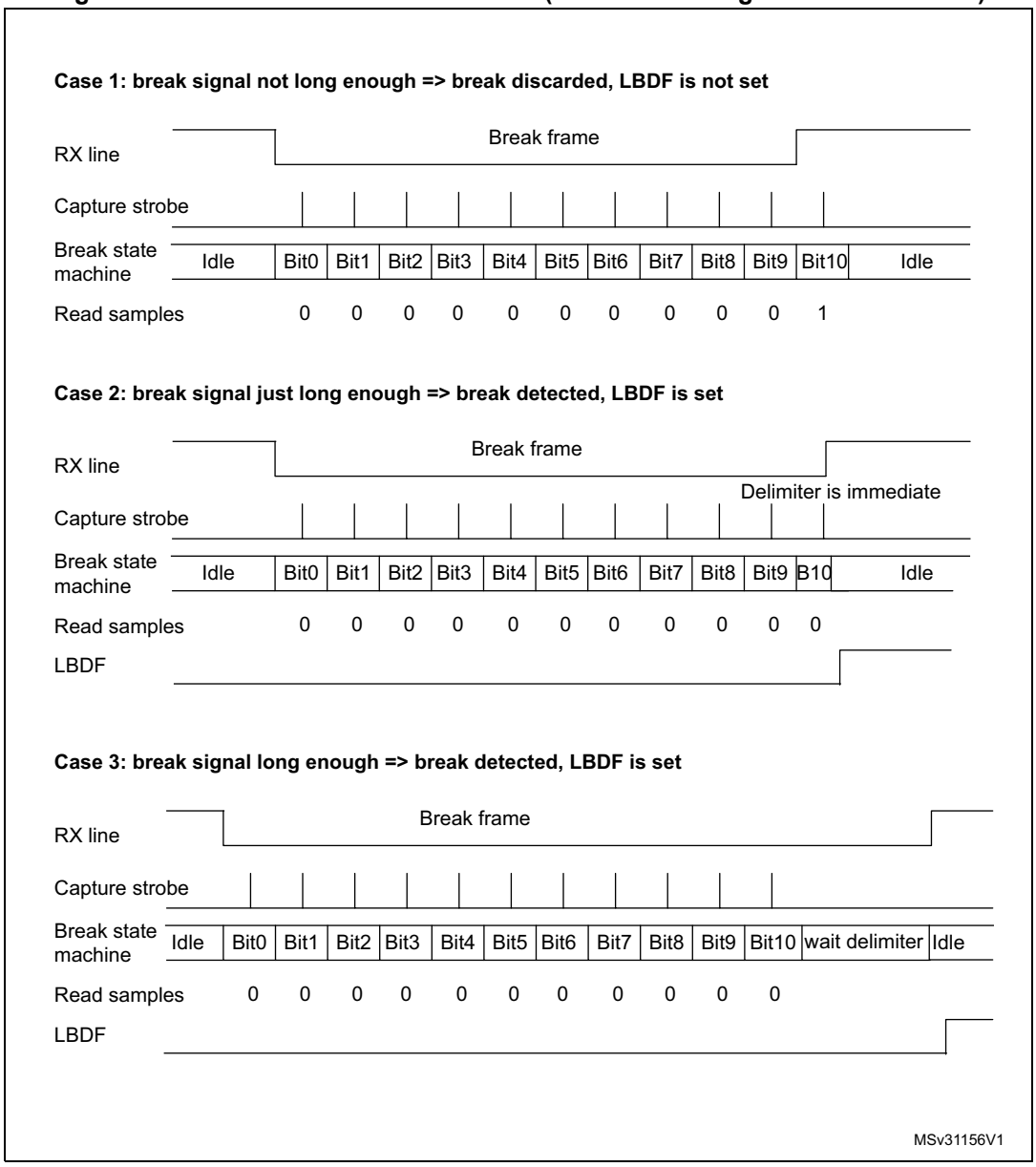
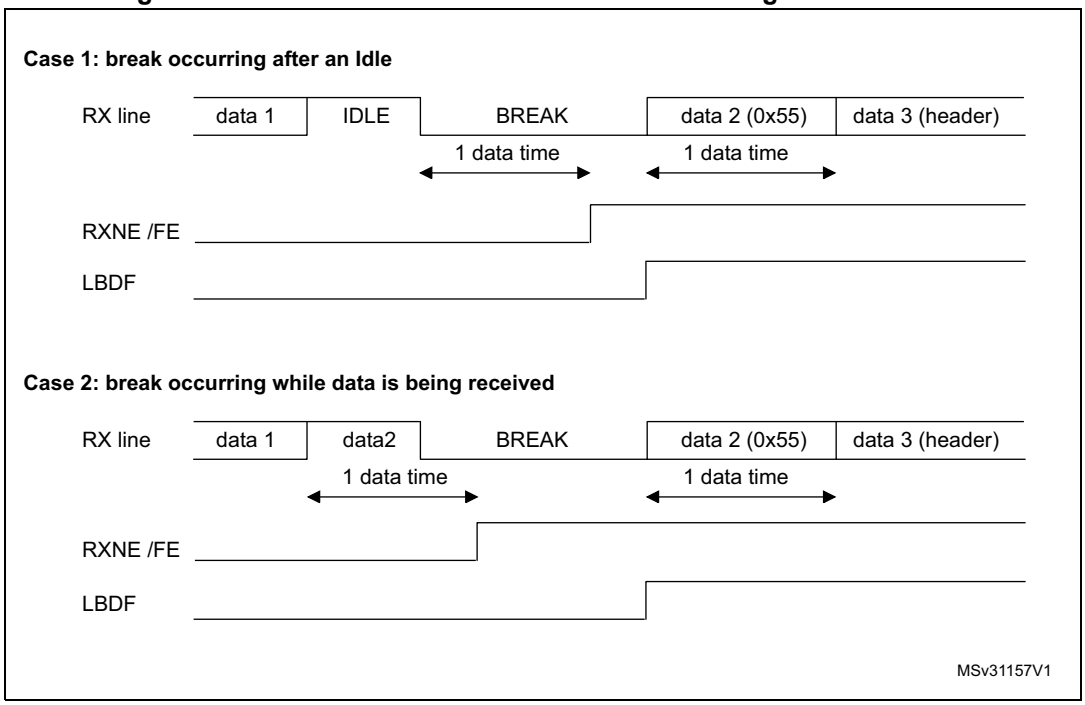


Figure 935. Break detection in LIN mode vs. Framing error detection



46.5.15 USART synchronous mode

Master mode

The synchronous master mode is selected by programming the CLKEN bit in the USART_CR2 register to 1. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see *Figure 936*, *Figure 937* and *Figure 938*).

During the Idle state, preamble and send break, the external CK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

Note: In master mode, the CK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE = 1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

Figure 936. USART example of synchronous master transmission

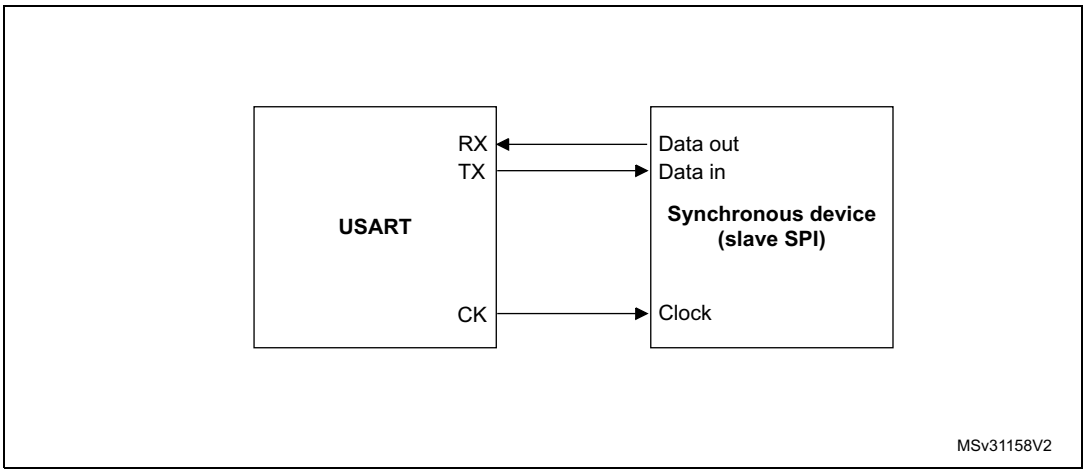


Figure 937. USART data clock timing diagram in synchronous master mode (M bits = 00)

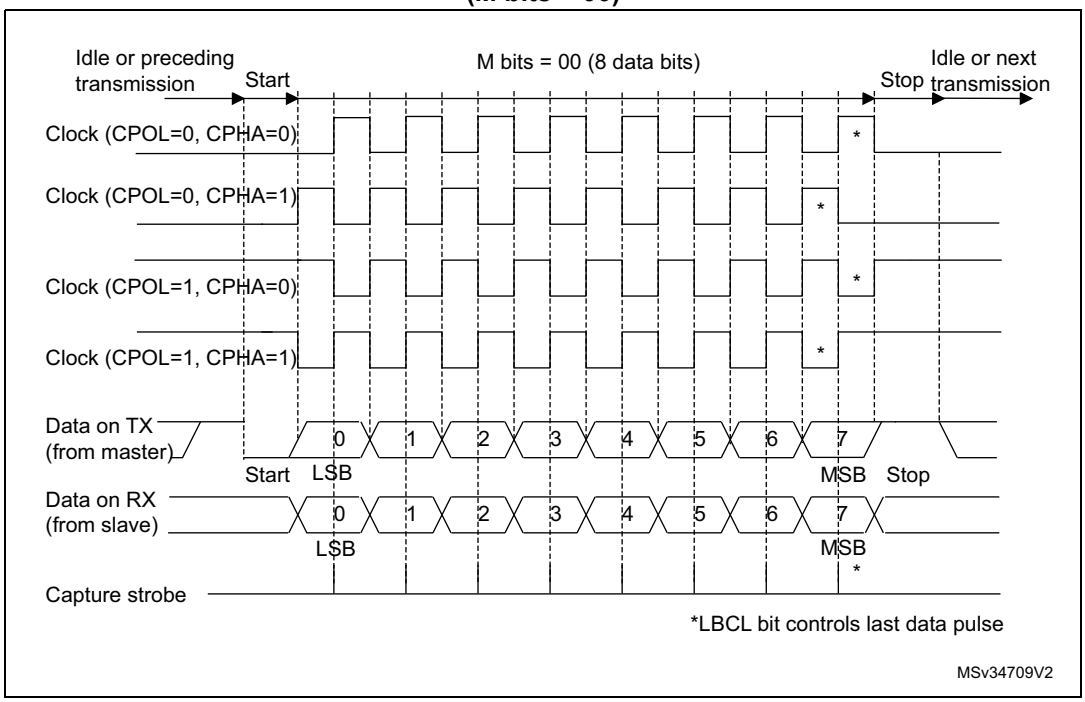
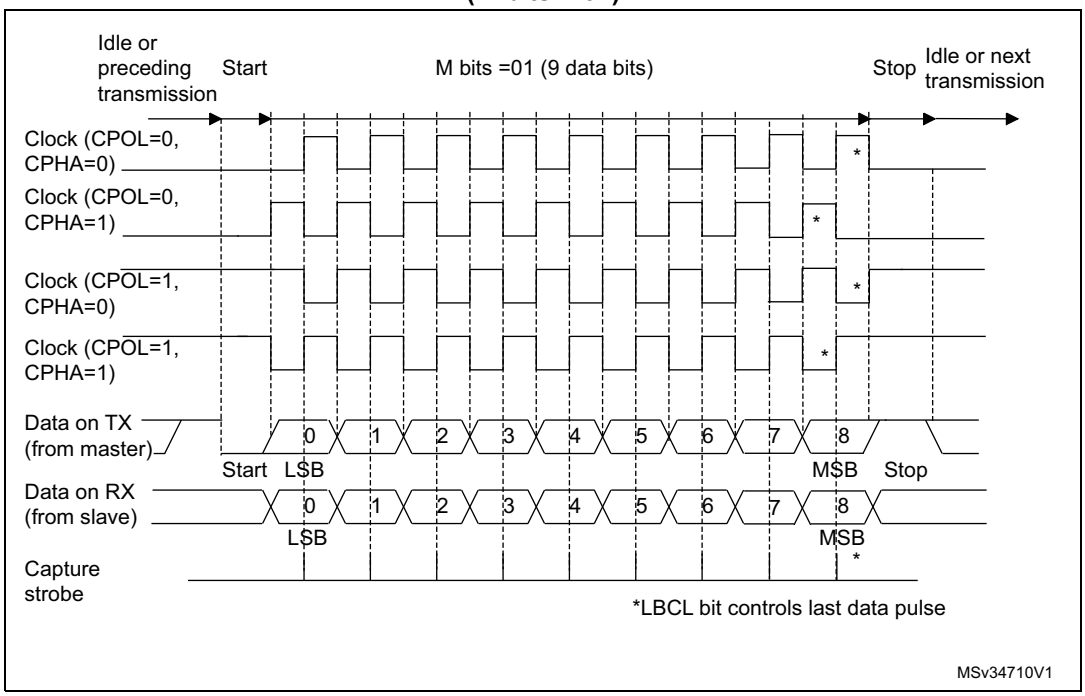


Figure 938. USART data clock timing diagram in synchronous master mode (M bits = 01)



Slave mode

The synchronous slave mode is selected by programming the SLVEN bit in the USART_CR2 register to 1. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The CK pin is the input of the USART in slave mode.

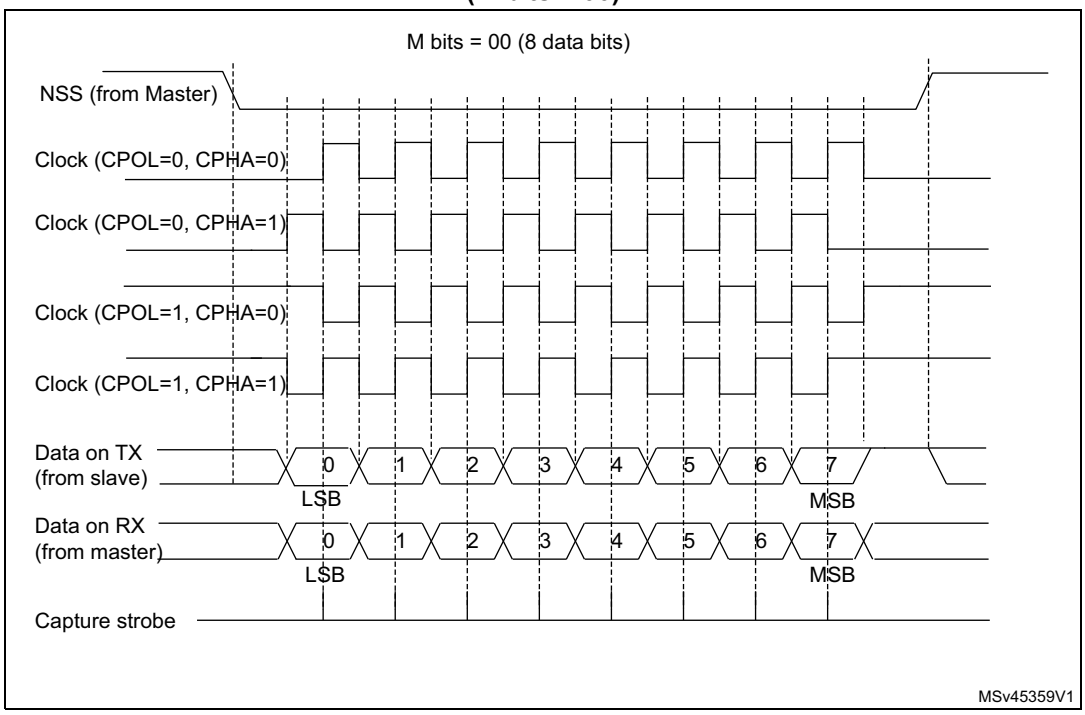
*Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (*usart_ker_ck_pres*) must be greater than 3 times the CK input frequency.*

The CPOL bit and the CPHA bit in the USART_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see *Figure 939*).

An underrun error flag is available in slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART_TDR.

The slave supports the hardware and software NSS management.

Figure 939. USART data clock timing diagram in synchronous slave mode (M bits = 00)



Slave Select (NSS) pin management

The hardware or software slave select management can be set through the DIS_NSS bit in the USART_CR2 register:

- Software NSS management (DIS_NSS = 1)

  SPI slave is always selected and NSS input pin is ignored.

  The external NSS pin remains free for other application uses.
- Hardware NSS management (DIS_NSS = 0)

  The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

Note: The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE = 0) to ensure that the clock pulses function correctly.

In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.

SPI slave underrun error

When an underrun error occurs, the UDR flag is set in the USART_ISR register, and the SPI slave goes on sending the last data until the underrrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.

In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error enables sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

Note: An underrun error may occur if the moment the data is written to the USART_TDR is too close to the first CK transmission edge. To avoid this underrun error, the USART_TDR must be written 3 usart_ker_ck *cycles before the first CK edge.*

46.5.16 USART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

46.5.17 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = 00 or STOP = 11
- from the end of the second stop bit if STOP = 10.
- from the beginning of the stop bit if STOP = 01.

When the timeout duration has elapsed, the RTOF flag in the USART_ISR register is set. A timeout is generated if RTOIE bit in USART_CR1 register is set.

46.5.18 USART smartcard mode

This section is relevant only when smartcard mode is supported. Refer to *Section 46.4: USART implementation*.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

The CLKEN bit can also be set to provide a clock to the smartcard.

The smartcard interface is designed to support asynchronous smartcard protocol as defined in the ISO 7816-3 standard. Both T = 0 (character mode) and T = 1 (block mode) are supported.

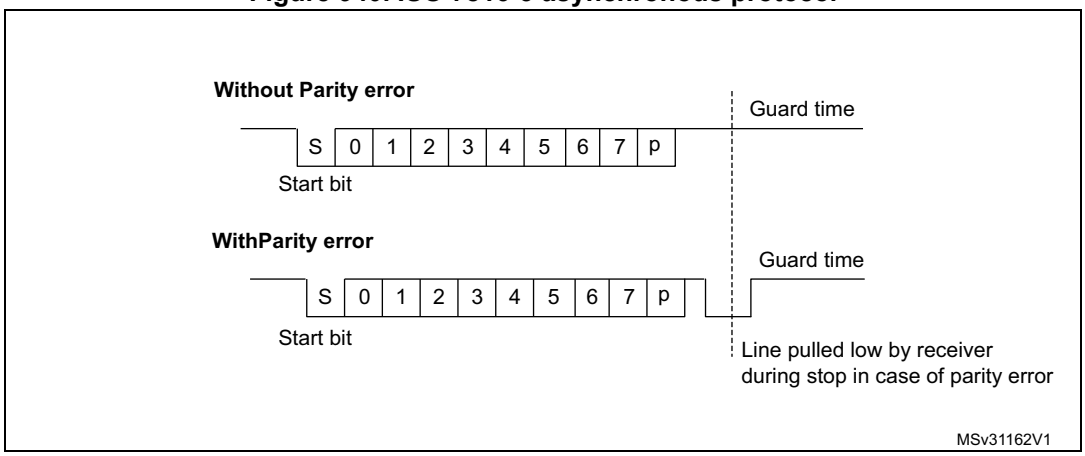
The USART must be configured as:

- 8 bits plus parity: M = 1 and PCE = 1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving data: STOP = 11 in the USART_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In T = 0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

Figure 940 shows examples of what can be seen on the data line with and without parity error.

Figure 940. ISO 7816-3 asynchronous protocol



When connected to a smartcard, the TX output of the USART drives a bidirectional line that is also driven by the smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.

The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART_RQR register.

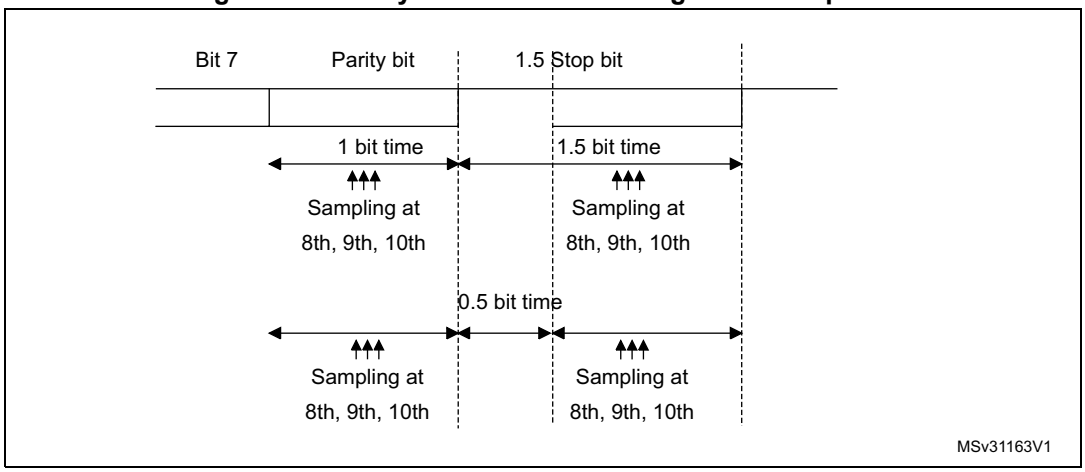
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guard time). If the software wants to repeat it again, it must ensure the minimum 2-baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T = 1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn't repeat the character.
- In transmission, the USART inserts the guard time (as programmed in the guard time register) between two successive characters. As the guard time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT character guard time, as defined by the 7816-3 specification) minus 12 (the duration of one character).
- The assertion of the TC flag can be delayed by programming the guard time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In smartcard mode an empty transmit shift register triggers the guard time counter to count up to the programmed value in the guard time register. TC is forced low during this time. When the guard time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.
- The de-assertion of TC flag is unaffected by smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note: Break characters are not significant in smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 941 shows how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 941. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the smartcard through the CK output. In smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART_GTPR register. CK frequency can be programmed from usart_ker_ck_pres/2 to usart_ker_ck_pres/62, where usart_ker_ck_pres is the peripheral input clock divided by a programmed prescaler.

Block mode (T = 1)

In T = 1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the USART_CR3 register.

When requesting a read from the smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) – 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

Note: The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time – 11 value), in order to enable the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: As in the smartcard protocol definition, the BWT/CWT values must be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT – 11 or CWT – 11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value

(0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value is programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

Note: The error checking code (LRC/CRC) must be computed/verified by software.

Direct and inverse convention

The smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 0, DATAINV = 0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 1, DATAINV =1.

Note: When logical data values are inverted (0 = H, 1 = L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH ≥ the USART received character is equal to 03 and the parity is odd.

Therefore, two methods are available for TS pattern recognition:

Method 1

The USART is programmed in standard smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHL LLL LLH = 0x103: inverse convention to be chosen

(H) LHHL HHH LLH = 0x13B: direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

46.5.19 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 46.4: USART implementation*.

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies the use of a return-to-zero, inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 942*).

The encoding and decoding of data in IrDA mode is handled by hardware blocks.

Even if IrDA is a half-duplex protocol, it requires two pins for transmission and reception (USART_TX and USART_RX).

The SIR transmit encoder modulates the non return-to-zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 kbauds for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is

ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not encoded. While receiving data, transmission must be avoided as the data to be transmitted may be corrupted.

- A 0 is transmitted as a high pulse and a 1 is transmitted as a 0. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see *Figure 943*).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than two periods, where the period equals PSC / usart_ker_ck_pres (PSC being the prescaler value programmed in the PSC[7:0] bitfield of the USART_GTPR register). Pulses of width less than one period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than two periods are accepted as a pulse. The IrDA encoder/decoder does not work when PSC = 0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART_CR2 register must be configured to 1 stop bit.

IrDA low-power mode

- Transmitter

  In low-power mode, the pulse width is not maintained at 3 / 16 of the bit period. Instead, the width of the pulse is three times the IrDA low-power period. The IrDA low-power frequency minimum value is 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < IrDA low-power frequency < 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.
- Receiver

  Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART must discard pulses of duration shorter than one IrDA low-power period. A valid low is accepted only if its duration is greater than two IrDA low-power periods.

Note: IrDA low-power period = PSC / usart_ker_ck_pres, where PSC corresponds to the value programmed in the PSC[7:0] bitfield of *the USART_GTPR register.*

A pulse of width less than two and greater than one IrDA low-power period may or may not be rejected.

The receiver set-up time must be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception.

Figure 942. IrDA SIR ENDEC block diagram

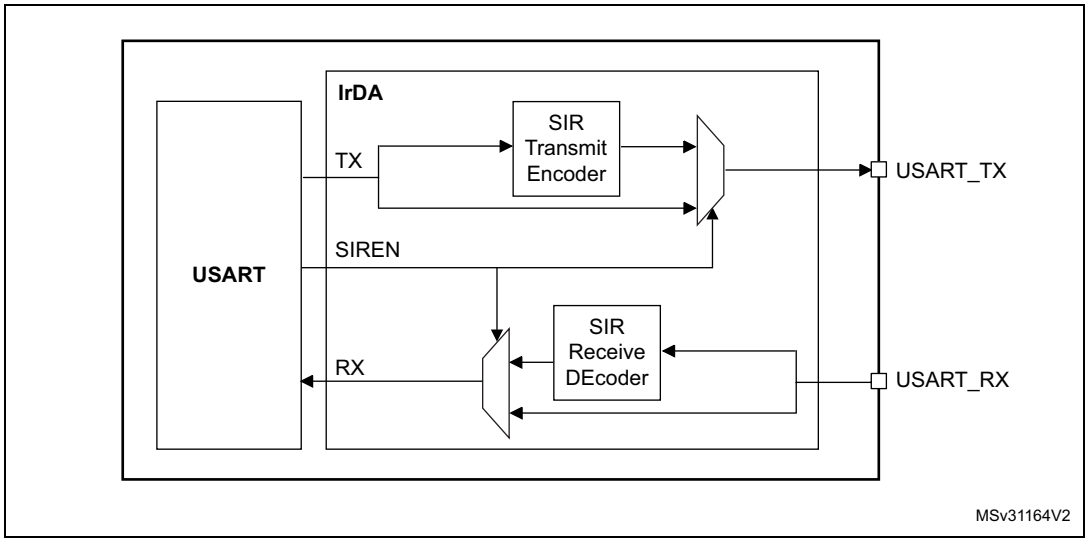
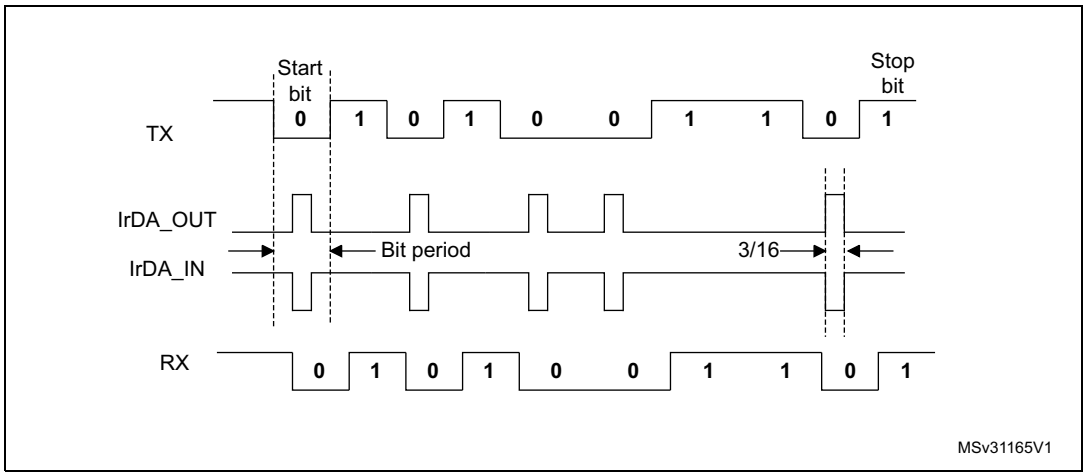


Figure 943. IrDA data modulation (3/16) - normal mode



46.5.20 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 46.4: USART implementation to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 46.5.7. To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

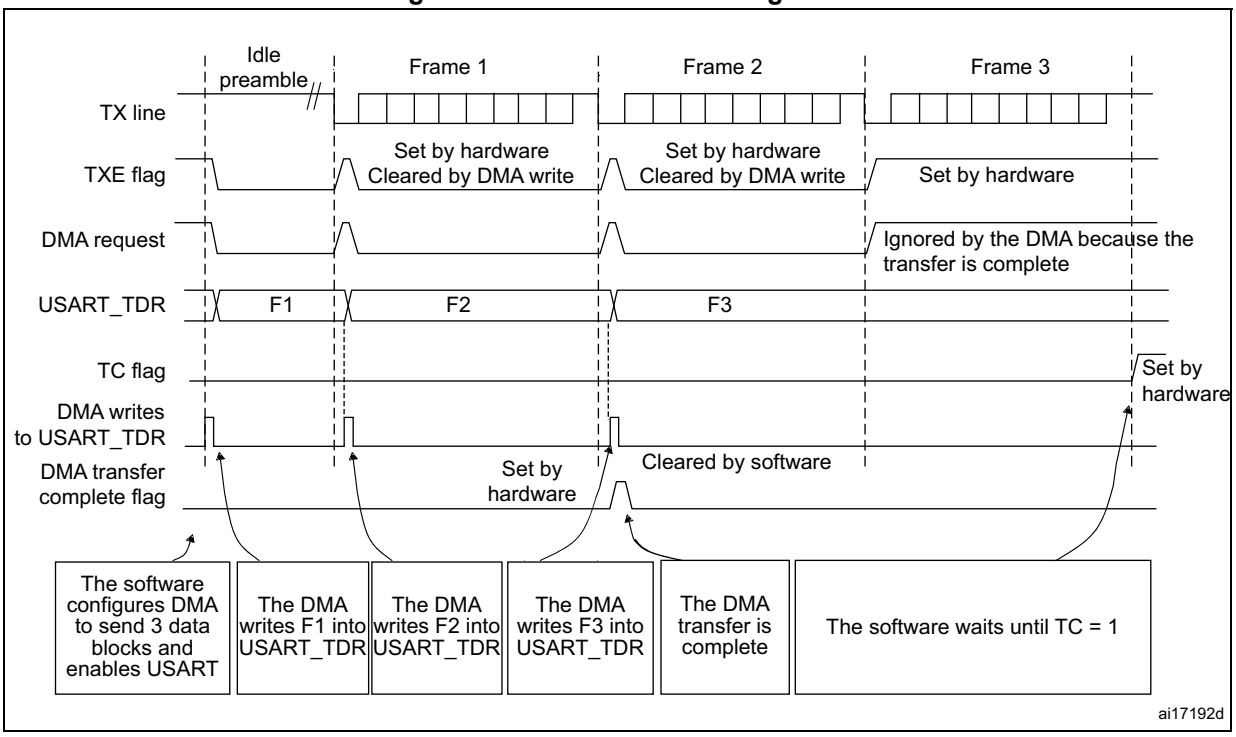
1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (DMA transfer complete), the TC flag can be monitored to make sure that the USART communication has completed. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 944. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

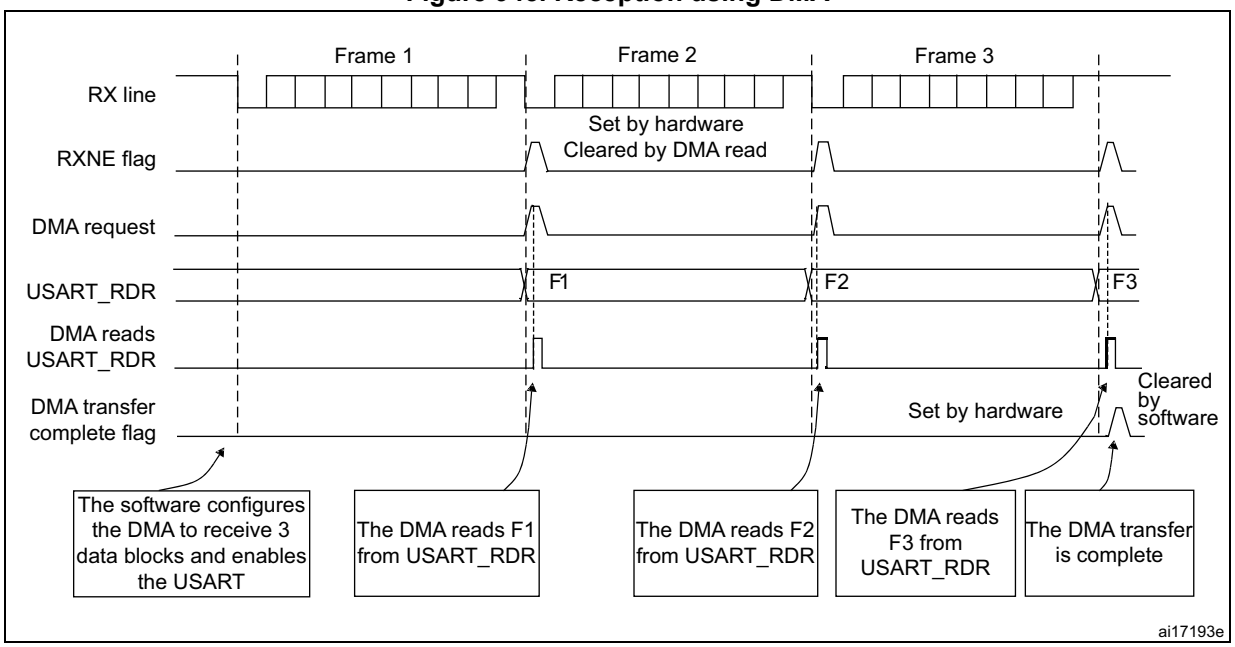
DMA mode can be enabled for reception by setting the DMAR bit in the USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 945. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

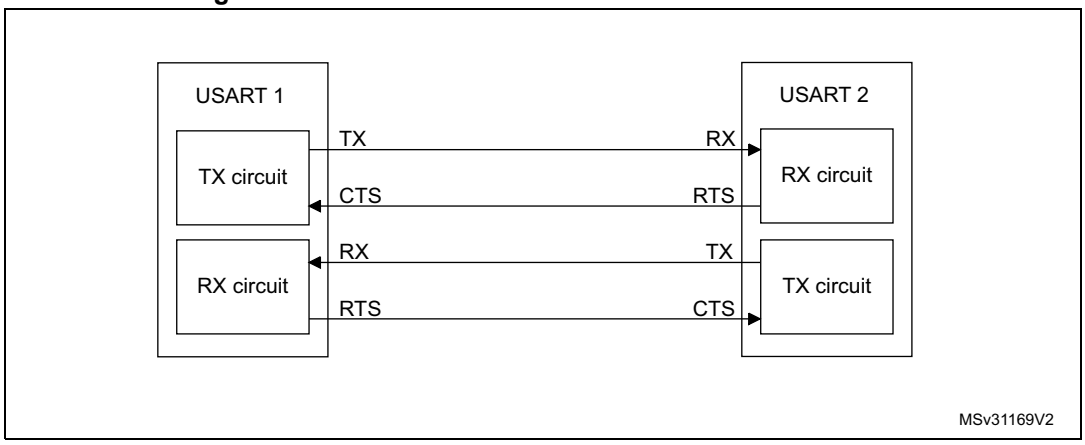
Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag, which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

46.5.21 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. The *Figure 946* shows how to connect two devices in this mode:

Figure 946. Hardware flow control between two USARTs

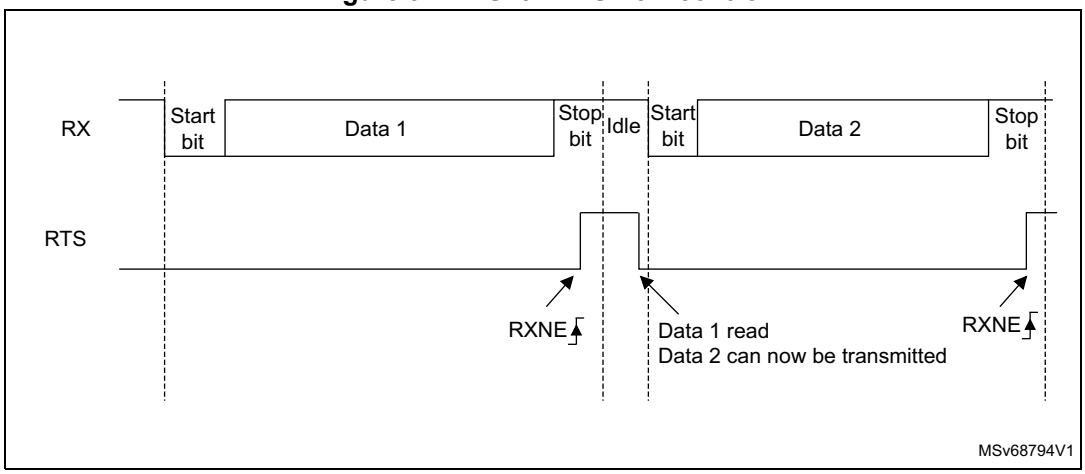


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to 1 in the USART_CR3 register.

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 947* shows an example of communication with RTS flow control enabled.

Figure 947. RS232 RTS flow control



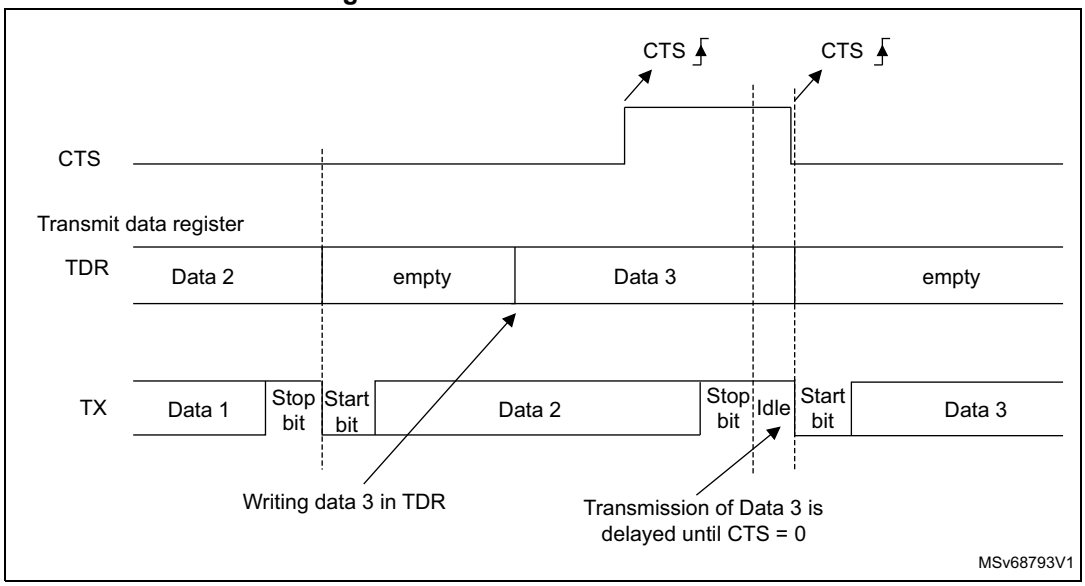
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 948* shows an example of communication with CTS flow control enabled.

Figure 948. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least three USART clock source periods before the end of the current character. In addition, it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This enables the user to activate the external transceiver control, through the DE (driver enable) signal. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

46.5.22 USART low-power management

The USART has advanced low-power mode functions, that enables transferring properly data even when the usart_pclk clock is disabled.

The USART is able to wake up the MCU from low-power mode when the UESM bit is set.

When the usart_pclk is gated, the USART provides a wake-up interrupt (**usart_wkup**) if a specific action requiring the activation of the **usart_pclk** clock is needed:

- If FIFO mode is disabled

  usart_pclk clock has to be activated to empty the USART data register.

  In this case, the usart_wkup interrupt source is RXNE set to 1. The RXNEIE bit must be set before entering low-power mode.
- If FIFO mode is enabled

  usart_pclk clock has to be activated to:
  - To fill the TXFIFO, or
  - To empty the RXFIFO

  In this case, the usart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the usart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time required to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many USART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **usart_wkup** interrupt can be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and a **usart_wkup** interrupt is generated if the WUFIE bit is set.

Note: Before entering low-power mode, make sure that no USART transfers are ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the USART is enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and reenabled when exiting from low-power mode.

When the FIFO is enabled, waking up from low-power mode on address match is only possible when mute mode is enabled.

Using mute mode with low-power mode

If the USART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, mute mode can be used with wake-up from low-power mode without any constraints (that is, the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wake-up from low-power mode when USART kernel clock (usart_ker_ck) is OFF in low-power mode

If during low-power mode, the usart_ker_ck clock is switched OFF when a falling edge on the USART receive line is detected, the USART interface requests the usart_ker_ck clock to be switched ON thanks to the usart_ker_ck_req signal. usart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, usart_ker_ck is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.

Figure 949 shows the USART behavior when the wake-up event is verified.

Figure 949. Wake-up event verified (wake-up event = address match, FIFO disabled)

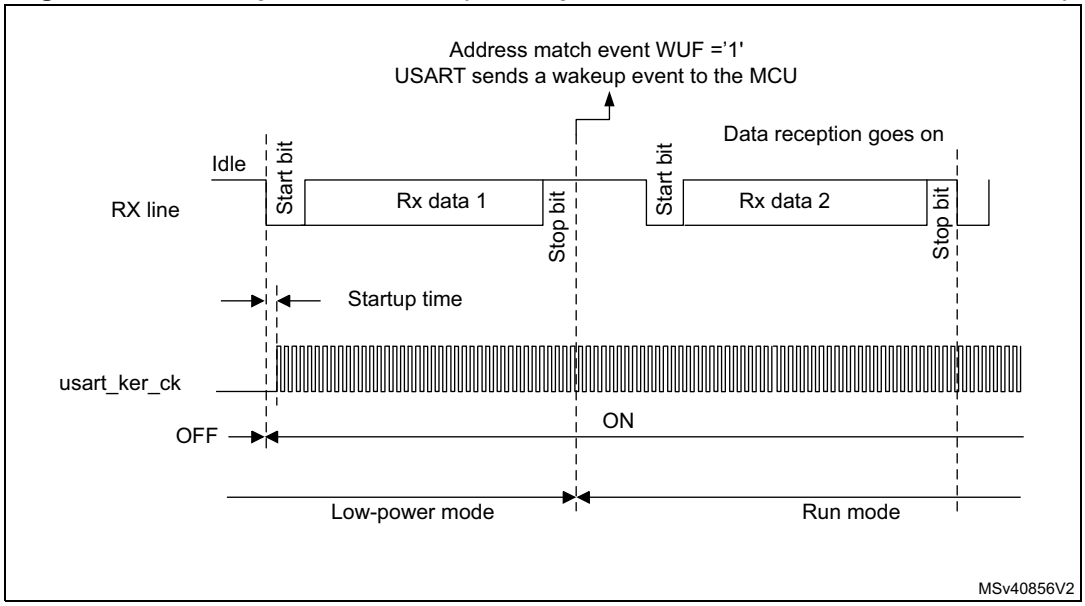
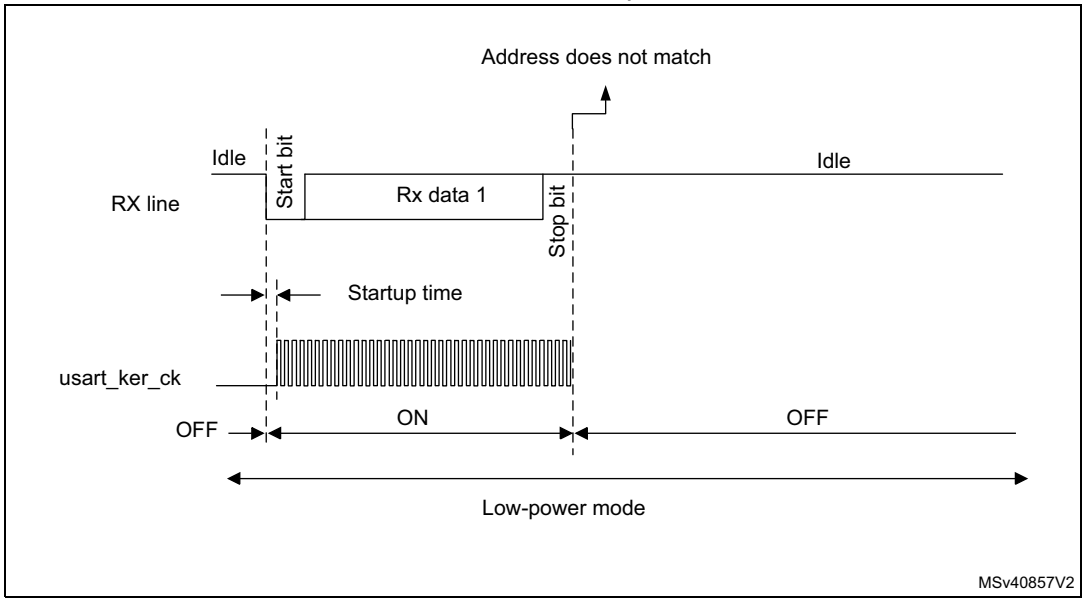


Figure 950 shows the USART behavior when the wake-up event is not verified.

Figure 950. Wake-up event not verified (wake-up event = address match, FIFO disabled)



Note: The figures above are valid when address match or any received frame is used as a wake-up event. If the wake-up event is the start bit detection, the USART sends the wake-up event to the MCU at the end of the start bit.

Determining the maximum USART baud rate that enables to correctly wake up the microcontroller from low-power mode

The maximum baud rate that enables to correctly wake up the microcontroller from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the USART receiver tolerance (see *Section 46.5.9: Tolerance of the USART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 537: Tolerance of the USART receiver when BRR [3:0] = 0000*, the USART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < USART receiver tolerance

$D_{WUmax} = t_{WUUSART} / (11 \times T_{bitmin})$

$T_{bitmin} = t_{WUUSART} / (11 \times D_{WUmax})$

where $t_{WUUSART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In fact, we need to consider at least the usart_ker_ck inaccuracy (DREC).

For example, if HSI is used as usart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WUUSART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = USART receiver tolerance – DREC = 3.41% – 1% = 2.41%

T_{bitmin} = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

46.6 USART in low-power modes

Table 540. Effect of low-power modes on the USART

Mode	Description
Sleep	No effect. USART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the USART registers is kept. The USART is able to wake up the microcontroller from Stop mode when the USART is clocked by an oscillator available in Stop mode.
Standby	The USART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 46.4: USART implementation* to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

46.7 USART interrupts

Refer to *Table 541* for a detailed description of all USART interrupt requests.

Table 541. USART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
USART or UART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	No	No
	Transmit FIFO not full	TXFNF	TXFNFIE	TXFIFO full		No	
	Transmit FIFO empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR(2)		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission complete	TC	TCIE	Write TDR or write 1 in TCCF		No	
	Transmission complete before guard time	TCBGT	TCBGTIE	Write TDR or write 1 in TCBGT		No	

Table 541. USART interrupt requests (continued)

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
USART or UART	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	No
	Receive FIFO not empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO full	RXFF(3)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX FNEIE	Write 1 in ORECF		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
	LIN break	LBDF	LBDIE	Write 1 in LBDCF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		No	
	Overrun error in multibuffer communication.	ORE(4)		Write 1 in ORECF		No	
	Framing error in multibuffer communication.	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Receiver timeout	RTOF	RTOFIE	Write 1 in RTOCCF		No	
	End of Block	EOBF	EOBIE	Write 1 in EOBCF		No	
	Wake-up from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	
	SPI slave underrun error	UDR	EIE	Write 1 in UDRCF		No	

1. The USART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to *Section 46.4: USART implementation* for the list of supported Stop modes.
2. Writing to TDR clears the TXFT flag only if the number of empty locations is less than the value programmed in TXFTCFG[2:0].
3. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART_RDR. In Stop mode, USART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
4. When OVRDIS = 0.

46.8 USART registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

46.8.1 USART control register 1 (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enable, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE**: RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when RXFF = 1 in the USART_ISR register

Bit 30 **TXFEIE**: TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFE = 1 in the USART_ISR register

Bit 29 **FIFOEN**: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
M[1:0] = 00: 1 start bit, 8 Data bits, n stop bits
M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits
M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits
This bit can only be written when the USART is disabled (UE = 0).
Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 46.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the USART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TXFNF = 1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever ORE = 1 or RXFNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the USART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

46.8.2 USART control register 1 [alternate] (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN**: FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits

M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits

This bit can only be written when the USART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 46.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the USART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TXE = 1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: USART interrupt is generated whenever ORE = 1 or RXNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the USART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

46.8.3 USART control register 2 (USART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

Bits 31:24 **ADD[7:0]**: Address of the USART node

These bits give the address of the USART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter is equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter is equal to 1.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the USART is disabled (UE = 0).

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bits 22:21 **ABRMOD[1:0]**: Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bitfield can only be written when ABREN = 0 or the USART is disabled (UE = 0).

Note: If DATAINV = 1 and/or MSBFIRST = 1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 20 **ABREN**: Auto baud rate enable

This bit is set and cleared by software.

0: Auto baud rate detection is disabled.

1: Auto baud rate detection is enabled.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1 = H, 0 = L)

1: Logical data from the data register are send/received in negative/inverse logic. (1 = L, 0 = H).

The parity bit is also inverted.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)

1: TX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).

This enables the use of an external inverter on the TX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)

1: RX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).

This enables the use of an external inverter on the RX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 14 **LINEN**: LIN mode enable

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bits 13:12 **STOP[1:0]**: Stop bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit.

10: 2 stop bits

11: 1.5 stop bits

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 11 **CLKEN**: Clock enable

This bit enables the user to enable the CK pin.

0: CK pin disabled

1: CK pin enabled

Note: If neither synchronous mode nor smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

In smartcard mode, in order to provide correctly the CK clock to the smartcard, the steps below must be respected:
UE = 0
SCEN = 1
GTPR configuration
CLKEN= 1
UE = 1

Bit 10 **CPOL**: Clock polarity

This bit enables the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window

1: Steady high value on CK pin outside transmission window

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 930* and *Figure 931*)

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the CK pin

1: The clock pulse of the last data bit is output to the CK pin

Caution: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).

0: Interrupt is inhibited

1: An interrupt is generated whenever LBDF = 1 in the USART_ISR register

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 5 **LBDL**: LIN break detection length

This bit is for selection between 11 bit or 10 bit break detection.

0: 10-bit break detection

1: 11-bit break detection

This bit can only be written when the USART is disabled (UE = 0).

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 4 **ADDM7**: 7-bit address detection/4-bit address detection

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection

1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the USART is disabled (UE = 0)

Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3 **DIS_NSS**:

When the DIS_NSS bit is set, the NSS pin input is ignored.

0: SPI slave selection depends on NSS input pin.

1: SPI slave is always selected and NSS input pin is ignored.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous slave mode enable
When the SLVEN bit is set, the synchronous slave mode is enabled.
0: Slave mode disabled.
1: Slave mode enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Note: The CPOL, CPHA, and LBCL bits must not be written while the transmitter is enabled.

46.8.4 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXF TIE	RXFTCFG[2:0]			TCBG TIE	TXFTIE	WUFIE	WUS1	WUS0	SCARCNT[2:0]			Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth
001:TXFIFO reaches 1/4 of its depth
010:TXFIFO reaches 1/2 of its depth
011:TXFIFO reaches 3/4 of its depth
100:TXFIFO reaches 7/8 of its depth
101:TXFIFO becomes empty
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG[2:0].

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth
001:Receive FIFO reaches 1/4 of its depth
010:Receive FIFO reaches 1/2 of its depth
011:Receive FIFO reaches 3/4 of its depth
100:Receive FIFO reaches 7/8 of its depth
101:Receive FIFO becomes full
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TCBGT = 1 in the USART_ISR register
Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG[2:0].

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever WUF = 1 in the USART_ISR register
Note: WUFIE must be set before entering in low-power mode.
If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count
This bitfield specifies the number of retries for transmission and reception in smartcard mode.
In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).
In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).
This bitfield must be programmed only when the USART is disabled (UE = 0).
When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.
0x0: retransmission disabled - No automatic retransmission in transmission mode.
0x1 to 0x7: number of automatic retransmission attempts (before signaling error)
Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the USART is disabled (UE = 0).
Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 46.4: USART implementation on page 2724.

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling smartcard mode.

0: Smartcard mode disabled

1: Smartcard mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1 or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

46.8.5 USART control register 3 [alternate] (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	WUFIE	WUS1	WUS0	SCARCNT[2:0]			Res.
							rw		rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TCBGT = 1 in the USART_ISR register

Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever WUF = 1 in the USART_ISR register

Note: WUFIE must be set before entering in low-power mode.

If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection

This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).

00: WUF active on address match (as defined by ADD[7:0] and ADDM7)

01: Reserved.

10: WUF active on start bit detection

11: WUF active on RXNE/RXFNE.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bitfield specifies the number of retries for transmission and reception in smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bitfield must be programmed only when the USART is disabled (UE = 0).

When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmission mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 46.4: USART implementation.

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling smartcard mode.

0: Smartcard mode disabled

1: Smartcard mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

46.8.6 USART baud rate register (USART_BRR)

This register can only be written when the USART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRR[15:0]:** USART baud rate

BRR[15:4]

BRR[15:4] correspond to USARTDIV[15:4]

BRR[3:0]

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].

When OVER8 = 1:

BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.

BRR[3] must be kept cleared.

46.8.7 USART guard time and prescaler register (USART_GTPR)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **GT[7:0]**: Guard time value

This bitfield is used to program the Guard time value in terms of number of baud clock periods.

This is used in smartcard mode. The transmission complete flag is set after this guard time value.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bits 7:0 **PSC[7:0]**: Prescaler value

Condition: IrDA low-power and normal IrDA mode

PSC[7:0] = IrDA normal and Low-power baud rate

This bitfield is used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

Condition: Smartcard mode

PSC[4:0]: Prescaler value

This bitfield is used for programming the prescaler for dividing the USART source clock to provide the smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

This bitfield can only be written when the USART is disabled (UE = 0).

Note: Bits [7:5] must be kept cleared if smartcard mode is used.

This bitfield is reserved and forced by hardware to 0 when the smartcard and IrDA modes are not supported. Refer to Section 46.4: USART implementation.

46.8.8 USART receiver timeout register (USART_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bits 31:24 **BLEN[7:0]**: Block Length

This bitfield gives the block length in smartcard T = 1 reception. Its value equals the number of information characters + the length of the Epilogue Field (1 – LEC / 2 – CRC) – 1.

Examples:

BLEN = 0 -> 0 information characters + LEC

BLEN = 1 -> 0 information characters + CRC

BLEN = 255 -> 254 information characters + CRC (total 256 characters))

In smartcard mode, the block length counter is reset when TXE = 0 (TXFE = 0 in case FIFO mode is enabled).

This bitfield can be used also in other modes. In this case, the block length counter is reset when RE = 0 (receiver disabled) and/or when the EOBCF bit is written to 1.

Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bitfield gives the Receiver timeout value in terms of number of bit duration.

In Standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In smartcard mode, this value is used to implement the CWT and BWT. See smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character.

Note: This value must only be programmed once per received character.

Note: RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to "0x00000000" when the receiver timeout feature is not supported. Refer to Section 46.4: USART implementation.

46.8.9 USART request register (USART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

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

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

Bit 4 **TXFRQ**: Transmit data flush request

When FIFO mode is disabled, writing 1 to this bit sets the TXE flag. This enables to discard the transmit data. This bit must be used only in smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register. If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART and smartcard modes.

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit empties the entire receive FIFO, that is clears the bit RXFNE.

This enables to discard the received data without reading them, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the USART in mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 **ABRRQ**: Auto baud rate request

Writing 1 to this bit resets the ABRF and ABRE flags in the USART_ISR and requests an automatic baud rate measurement on the next received data frame.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

46.8.10 USART interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of USART_CR3 register.
An interrupt is generated if the TXFTIE bit (bit 31) is set in the USART_CR3 register.
0: TXFIFO does not reach the programmed threshold.
1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the threshold programmed in the RXFTCFG[2:0] bitfield of the USART_CR3 register is reached. This means that there are (RXFTCFG[2:0] – 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE bit = 1 (bit 28) in the USART_CR3 register.
0: Receive FIFO does not reach the programmed threshold.
1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the USART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th data.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.
It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.
This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.
0: Transmission has not completed, or transmission has completed unsuccessfully (that is, a NACK is received from the card)
1: Transmission has completed successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 46.4: USART implementation.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART_RDR register.
An interrupt is generated if the RXFFIE bit = 1 in the USART_CR1 register.
0: RXFIFO not full.
1: RXFIFO Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register.
An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the USART_CR1 register.
0: TXFIFO not empty.
1: TXFIFO empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 46.4: USART implementation.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 46.4: USART implementation.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 46.4: USART implementation.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1in the USART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception ongoing

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation has completed without success (ABRE = 1) (ABRE, RXFNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 46.4: USART implementation.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to EOBCF in the USART_ICR register.

0: End of block not reached

1: End of block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 46.4: USART implementation.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 46.4: USART implementation.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART_TDR. Every write operation to the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART_TDR.

An interrupt is generated if the TXFNFIE bit = 1 in the USART_CR1 register.

0: Transmit FIFO is full

1: Transmit FIFO is not full

Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE is set at the same time).

This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data has completed, and the TXFE bit is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the USART_ICR register, or by a write to the USART_TDR register.

0: Transmission has not completed

1: Transmission has completed

Note: If the TE bit is reset and no transmission is ongoing, the TC bit is immediately set.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO.

RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 46.5.9: Tolerance of the USART receiver to clock deviation).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the USART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.
An interrupt is generated if PEIE = 1 in the USART_CR1 register.
0: No parity error
1: Parity error

Note: This error is associated with the character in the USART_RDR.

46.8.11 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported)

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
						r			r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.

It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed, and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission has not completed or transmission has completed unsuccessfully (that is, a NACK is received from the card)

1: Transmission has not completed successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 46.4: USART implementation.

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 46.4: USART implementation.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 46.4: USART implementation.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 46.4: USART implementation.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1 in the USART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception ongoing

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation has completed without success (ABRE = 1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 46.4: USART implementation.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to EOBCF in the USART_ICR register.

0: End of block not reached

1: End of block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 46.4: USART implementation.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 46.4: USART implementation.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by writing to the USART_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in smartcard T = 0 mode, in case of transmission failure).

An interrupt is generated if the TXEIE bit = 1 in the USART_CR1 register.

0: Data register full

1: Data register empty

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the USART_ICR register or by writing to the USART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the USART_RDR shift register has been transferred to the USART_RDR register. It is cleared by reading from the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXNE = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 46.5.9: Tolerance of the USART receiver to clock deviation).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the USART_RDR.

46.8.12 USART interrupt flag clear register (USART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
		w	w	w		w	w	w	w	w	w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the USART_ISR register.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the USART_ISR register.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRCF**:SPI slave underrun clear flag

Writing 1 to this bit clears the UDRF flag in the USART_ISR register.

Note: If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation

Bit 12 **EOBCF**: End of block clear flag

Writing 1 to this bit clears the EOBF flag in the USART_ISR register.

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 11 **RTOCF**: Receiver timeout clear flag
Writing 1 to this bit clears the RTOF flag in the USART_ISR register.
Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.
Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 8 **LBDCF**: LIN break detection clear flag
Writing 1 to this bit clears the LBDF flag in the USART_ISR register.
Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag
Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag
Writing 1 to this bit clears the TXFE flag in the USART_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the USART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the USART_ISR register.

46.8.13 USART receive data register (USART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value

Contains the received data character.

The RDR register provides the parallel interface between the input shift register and the internal bus (see *Section 46.5.1: USART block diagram*).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

46.8.14 USART transmit data register (USART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

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

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

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The USART_TDR register provides the parallel interface between the internal bus and the output shift register (see *Section 46.5.1: USART block diagram*).

When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

46.8.15 USART prescaler register (USART_PRESC)

This register can only be written when the USART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

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

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

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

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The USART input clock can be divided by a prescaler factor:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Others: Reserved, must not be used

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011 that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 46.4: USART implementation.

46.8.16 USART register map

Table 542. USART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_CR1 FIFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	USART_CR1 FIFO mode disabled	Res.	Res.	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USART_CR2	ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0			0
0x08	USART_CR3 FIFO mode enabled	TXFTCFG[2:0]			RXFTIE	RXFTCFG2:0]			TCBGTIE	TXFTIE	WUFIE	WUS 1	WUS 0	SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	USART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	WUFIE	WUS 1	WUS 0	SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value								0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 542. USART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	USART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_ GTPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USART_ RTOR	BLEN[7:0]								RTO[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
	Reset value																												0	0	0	0	0
0x1C	USART_ISR FIFO mode enabled	Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value					0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x1C	USART_ISR FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value							0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
	Reset value												0			0				0	0	0		0	0	0	0	0	0	0	0	0	0
0x24	USART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	USART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	USART_ PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

47 Low-power universal asynchronous receiver transmitter (LPUART)

47.1 LPUART introduction

The LPUART is an UART, which enables bidirectional UART communications with a limited power consumption. Only a 32.768 kHz LSE clock is required to enable UART communications up to 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports half-duplex single-wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.

47.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 bauds to 9600 bauds using a 32.768 kHz clock source.
- Higher baud rates can be achieved by using a higher frequency clock source
- Two internal FIFOs to transmit and receive data

  Each FIFO can be enabled/disabled by software and come with status flags for FIFOs states.
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK.
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS485 transceiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - Busy and end of transmission flags

- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise detection
  - Frame error
  - Parity error
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode

47.3 LPUART implementation

The tables below describe the LPUART implementation. USARTs and UARTs are included for comparison.

Table 543. Instance implementation on STM32H7Pxxx

Instance	STM32H7Pxxx
USART1	Full
USART2	Full
USART3	Full
USART6	Full
USART10	Full
UART4	Basic
UART5	Basic
UART7	Basic
UART8	Basic
UART9	Basic
LPUART1	Low-power

Table 544. USART/LPUART features

Modes/features(1)	Full feature	Basic feature	Low-power feature
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (master/slave)	X	-	-
Smartcard mode	X	-	-
Single-wire half-duplex communication	X	X	X

Table 544. USART/LPUART features (continued)

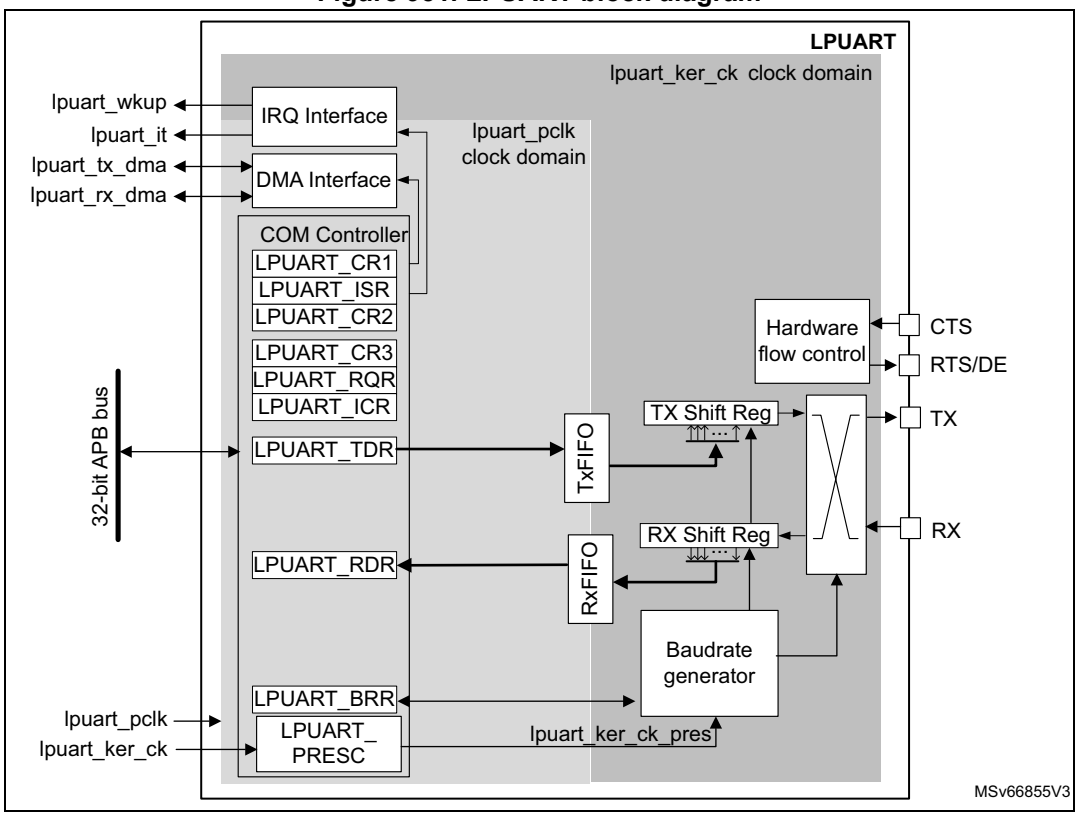
Modes/features[1]	Full feature	Basic feature	Low-power feature
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual clock domain	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	16		
Wake-up from low-power mode	X[2]	X[2]	X[2]

1. X = supported.
2. Wake-up supported from Stop mode.

47.4 LPUART functional description

47.4.1 LPUART block diagram

Figure 951. LPUART block diagram



47.4.2 LPUART pins and internal signals

Description LPUART input/output pins

- LPUART bidirectional communications

  LPUART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

  – **RX** (Receive Data Input):

  RX is the serial data input.

  – **TX (**Transmit Data Output)

  When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire mode, this I/O is used to transmit and receive the data.

- RS232 hardware flow control mode

  The RS232 hardware flow control mode requires the following pins:

  – **CTS (**Clear To Send)

  When driven high, this signal blocks the data transmission at the end of the current transfer.

  – **RTS** (Request to send)

  When it is low, this signal indicates that the LPUART is ready to receive data.

- RS485 hardware flow control mode

  The **DE** (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

Refer to *Table 545* and *Table 546* for the list of LPUART input/output pins and internal signals.

Table 545. LPUART input/output pins

Pin name	Signal type	Description
LPUART_RX	Input	Serial data receive input.
LPUART_TX	Output	Transmit data output.
LPUART_CTS	Input	Clear to send
LPUART_RTS	Output	Request to send
LPUART_DE(1)	Output	Driver enable

1. LPUART_DE and LPUART_RTS share the same pin.

Description LPUART input/output signals

Table 546. LPUART internal input/output signals

Signal name	Signal type	Description
lpuart_pclk	Input	APB clock
lpuart_ker_ck	Input	LPUART kernel clock
lpuart_wkup	Output	LPUART provides a wake-up interrupt

Table 546. LPUART internal input/output signals (continued)

Signal name	Signal type	Description
lpuart_it	Output	LPUART global interrupt
lpuart_tx_dma	Input/output	LPUART transmit DMA request
lpuart_rx_dma	Input/output	LPUART receive DMA request

47.4.3 LPUART clocks

The simplified block diagram given in *Section 47.4.1: LPUART block diagram* shows two fully independent clock domains:

- The **lpuart_pclk** clock domain

  The **lpuart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.
- The **lpuart_ker_ck** kernel clock domain

  The **lpuart_ker_ck** is the LPUART clock source. It is independent of the **lpuart_pclk** and delivered by the RCC. So, the LPUART registers can be written/read even when the **lpuart_ker_ck** is stopped.

  When the dual clock domain feature is not supported, the **lpuart_ker_ck i**s the same as the lpuart_pclk clock.

There is no constraint between **lpuart_pclk** and **lpuart_ker_ck**: **lpuart_ker_ck** can be faster or slower than **lpuart_pclk**, with no more limitation than the ability for the software to manage the communication fast enough.

47.4.4 LPUART character description

The word length can be set to 7 or 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART_CR1 register (see *Figure 925*).

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

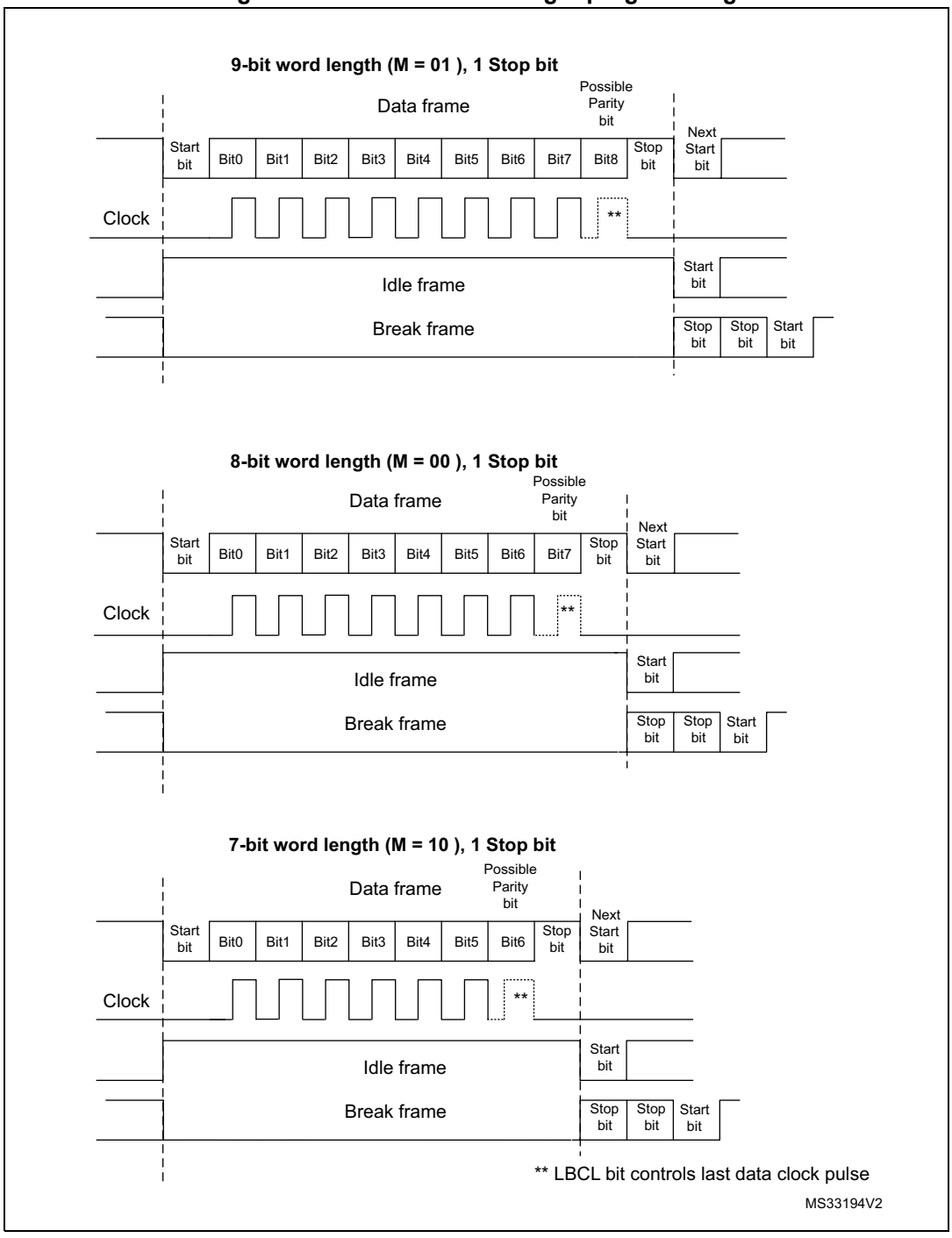
An ***idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.

Figure 952. LPUART word length programming



47.4.5 LPUART FIFOs and thresholds

The LPUART can operate in FIFO mode.

The LPUART comes with a transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN bit (bit 29) in the LPUART_CR1 register.

Since 9 bits the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However, the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the LPUART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield.

  In this case, the RXFT flag is set in the LPUART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in LPUART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when the RXFTCFG[2:0] is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received: FIFO size – 1 data in the RXFIFO and 1 data in the LPUART_RDR. As a result, the next received data does not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the LSART_CR3 register.

47.4.6 LPUART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The transmit enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

Character transmission

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Section 47.4.1: LPUART block diagram*).

When FIFO mode is enabled, the data written to the LPUART_TDR register are queued in the TXFIFO.

Every character is preceded by a start bit bit, which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be 1 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the LPUART_TDR.

The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost.

An idle frame is sent after the TE bit is enabled.

Configurable stop bits

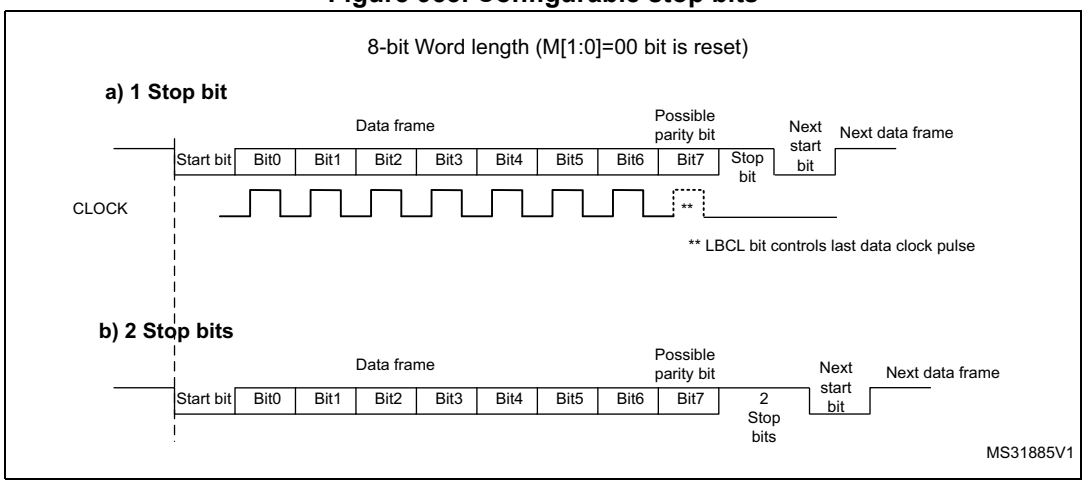
The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13, 12).

- ***1 stop bit***: This is the default value of the number of stop bits.
- ***2 stop bits***: This is supported by normal LPUART, single-wire, and modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 953. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAT) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 47.4.13: Continuous communication using DMA and LPUART*.
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.

7. Write the data to send in the LPUART_TDR register. Repeat this operation for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data in the LPUART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data in the LPUART_TDR adds one data to the TXFIFO. Write operations to the LPUART_TDR are performed when the TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the LPUART_TDR register, wait until TC = 1. This indicates that the transmission of the last frame has been completed.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame has been completed.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the LPUART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode disabled:

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware to indicate that:
  – the data have been moved from the LPUART_TDR register to the shift register and data transmission has started;
  – the LPUART_TDR register is empty;
  – the next data can be written to the LPUART_TDR register without overwriting the previous data.

  The TXE flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the LPUART_TDR register stores the data in the TDR register, which is copied to the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the LPUART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.
- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  – The TXFIFO is not full;
  – The LPUART_TDR register is empty;
  – The next data can be written to the LPUART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the

LPUART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

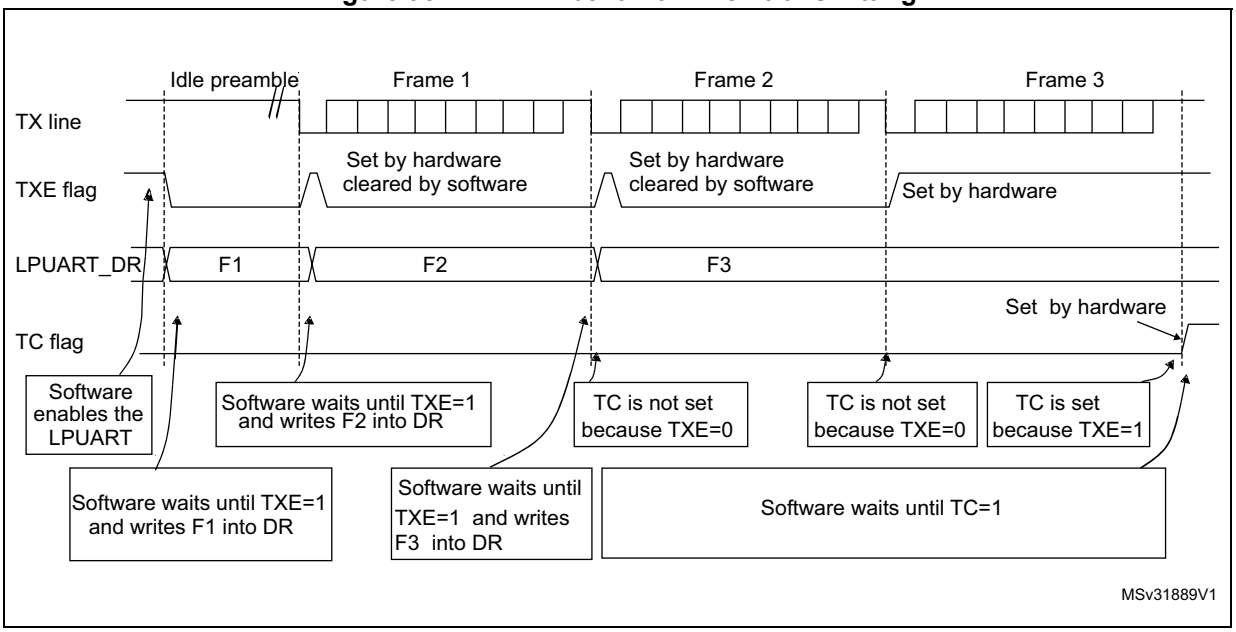
When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be written to the TXFIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed threshold.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART_CR1 register.

After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC = 1 before disabling the LPUART or causing the microcontroller to enter the low-power mode (see *Figure 954: TC/TXE behavior when transmitting*).

Figure 954. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see *Figure 952*).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

47.4.7 LPUART receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART_CR1 register.

Start bit detection

In the LPUART, the start bit is detected when a falling edge occurs on the Rx line, followed by a sample taken in the middle of the start bit to confirm that it is still 0. If the start sample is at 1, then the noise error flag (NE) is set, then the start bit is discarded and the receiver waits for a new start bit. Else, the receiver continues to sample all incoming bits normally.

Character reception

During an LPUART reception, data are shifted with the least significant bit first (default configuration) through the RX pin. In this mode, the LPUART_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART_BRR
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAR) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 47.4.13: Continuous communication using DMA and LPUART*.
6. Set the RE bit LPUART_CR1. This enables the receiver, which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. Reading the LPUART_RDR returns the oldest data entered in the RXFIFO.

When a data is received, it is stored in the RXFIFO, together with the corresponding error bits.

- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise, or an overrun error has been detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the receive data register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty, that is, there is a data in the RXFIFO to be read.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the LPUART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs when a character is received when RXNE has not been reset.

  Data cannot be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.

  An overrun error occurs if the RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – The ORE bit is set;
  – The RDR content is not lost. The previous data is available when a read to LPUART_RDR is performed.
  – The shift register is overwritten. After that, any data received during overrun is lost.
  – An interrupt is generated if either the RXNEIE bit or EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full.

  Data cannot be transferred from the shift register to the LPUART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set;
  – The first entry in the RXFIFO is not lost. It is available when a read to LPUART_RDR is performed.
  – The shift register is overwritten. After that, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE bit or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. T

When the FIFO mode is disabled, there are two possibilities

- *If RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *If RXNE = 0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.*

Selecting the clock source

The choice of the clock source is done through the clock control system (see *Section Reset and clock controller (RCC)*). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:

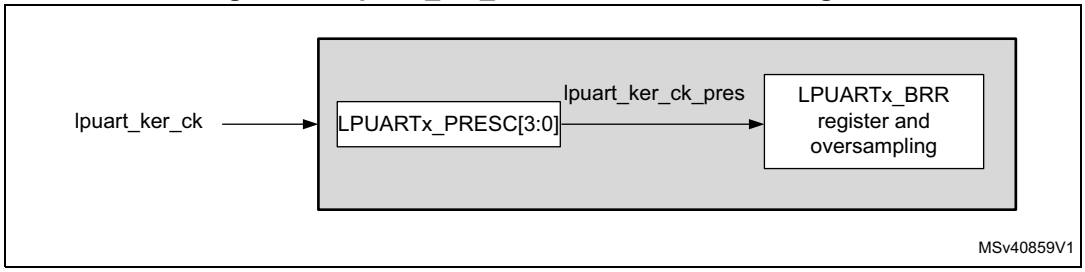
- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is lpuart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the lpuart_ker_ck clock source can be configured in the RCC (see *Section Reset and clock controller (RCC)*). Otherwise, the lpuart_ker_ck is the same as lpuart_pclk.

The lpuart_ker_ck can be divided by a programmable factor in the LPUART_PRESC register.

Figure 955. lpuart_ker_ck clock divider block diagram



Some lpuart_ker_ck sources enable the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active to enable LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming bit as close as possible to the middle of the bit-period. Only a single sample is taken of each of the incoming bits.

Note: There is no noise detection for data.

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a desynchronization or excessive noise.

When the framing error is detected:

- The FE bit is set by hardware.
- The invalid data is transferred from the Shift register to the LPUART_RDR register.
- No interrupt is generated in case of single byte communication. However, this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication, an interrupt is issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of LPUART_CR2: it can be either 1 or 2 in normal mode.

- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th, and 10th samples.
- **2 stop bits**: sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample, that is, during the second stop bit. The first stop bit is not checked for framing error.

47.4.8 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART_BRR register.

$$\text{Tx/Rx baud} = \frac{256 \times \text{lpuart_ker_ck_pres}}{\text{LPUARTDIV}}$$

LPUARTDIV is defined in the LPUART_BRR register.

Note: The baud counters are updated to the new value in the baud registers after a write operation to LPUART_BRR. Hence, the baud rate register value must not be changed during communication.

It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

lpuart_ker_ck_pres must range from 3 x baud rate to 4096 x baud rate.

The maximum baud rate that can be reached when the LPUART clock source is the LSE, is 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Table 547. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz

Baud rate		lpuart_ker_ck_pres= 32.768 kHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated – Desired) B.rate / Desired B.rate
1	0.3 kbaud	300 baud	0x6D3A	0
2	0.6 kbaud	600 baud	0x369D	0
3	1200 bauds	1200.087 bauds	0x1B4E	0.007
4	2400 bauds	2400.17 bauds	0xDA7	0.007
5	4800 bauds	4801.72 bauds	0x6D3	0.035
6	9600 kbauds	9608.94 bauds	0x369	0.093

47.4.9 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes that contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers, which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WULPUART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WULPUART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WULPUART}}{9 \times Tbit}$$

$t_{WULPUART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 548*:

- Number of stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

Table 548. Tolerance of the LPUART receiver

M bits	768 < BRR < 1024	1024 < BRR < 2048	2048 < BRR < 4096	4096 ≤ BRR
8 bits (M = 00), 1 stop bit	1.82%	2.56%	3.90%	4.42%
9 bits (M = 01), 1 stop bit	1.69%	2.33%	2.53%	4.14%
7 bits (M = 10), 1 stop bit	2.08%	2.86%	4.35%	4.42%
8 bits (M = 00), 2 stop bits	2.08%	2.86%	4.35%	4.42%
9 bits (M = 01), 2 stop bits	1.82%	2.56%	3.90%	4.42%
7 bits (M = 10), 2 stop bits	2.34%	3.23%	4.92%	4.42%

Note: The data specified in Table 548 may slightly differ in the special case when the received frames contain some idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M = 01 or 9- bit times when M = 10).

47.4.10 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, with their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant LPUART service overhead for all nonaddressed receivers.

The nonaddressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the LPUART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two lpuart_ker_ck *cycles), otherwise mute mode might remain active.*

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in the LPUART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART_RQR register, under certain conditions.

The LPUART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the LPUART_CR1 register:

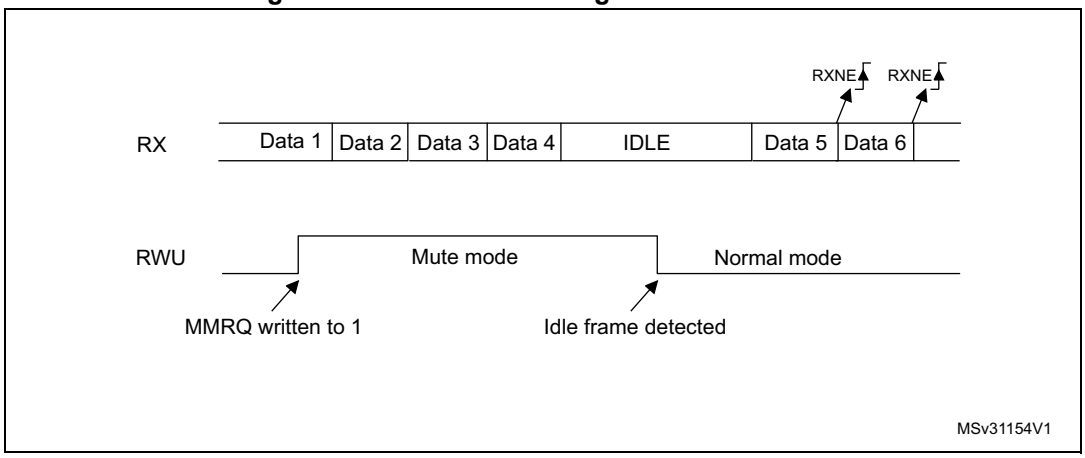
- Idle Line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The LPUART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The LPUART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the LPUART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 956*.

Figure 956. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a 1 otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7- or 4-bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address, which is programmed in the ADD bits in the LPUART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The LPUART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters mute mode.

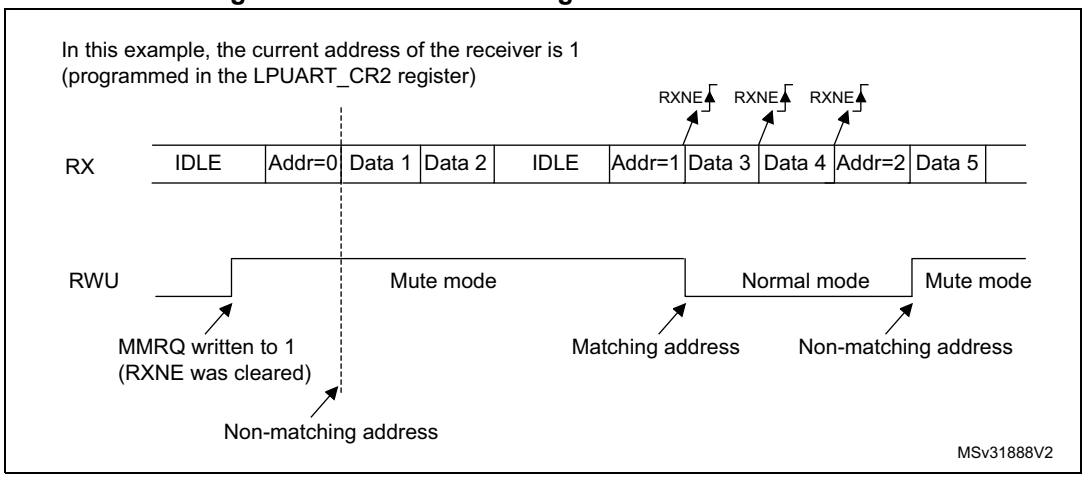
The LPUART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The LPUART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in mute mode.

An example of mute mode behavior using address mark detection is given in *Figure 957*.

Figure 957. Mute mode using address mark detection



47.4.11 LPUART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in *Table 549*.

Table 549: LPUART frame formats

M bits	PCE bit	LPUART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit.
2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame, which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in LPUART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in LPUART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART_ISR register and an interrupt is generated if PEIE is set in the LPUART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART_ICR register.

Parity generation in transmission

If the PCE bit is set in LPUART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

47.4.12 LPUART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the LPUART_CR3 register.

The LPUART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in LPUART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

Note: In LPUART communications, in the case of 1-stop bit configuration, the RXNE flag is set in the middle of the stop bit.

47.4.13 Continuous communication using DMA and LPUART

The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 47.3: LPUART implementation to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 47.4.7. To perform continuous communication. When FIFO is disabled, clear the TXE/ RXNE flags in the LPUART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAT bit in the LPUART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to Section *direct memory access controller*) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):

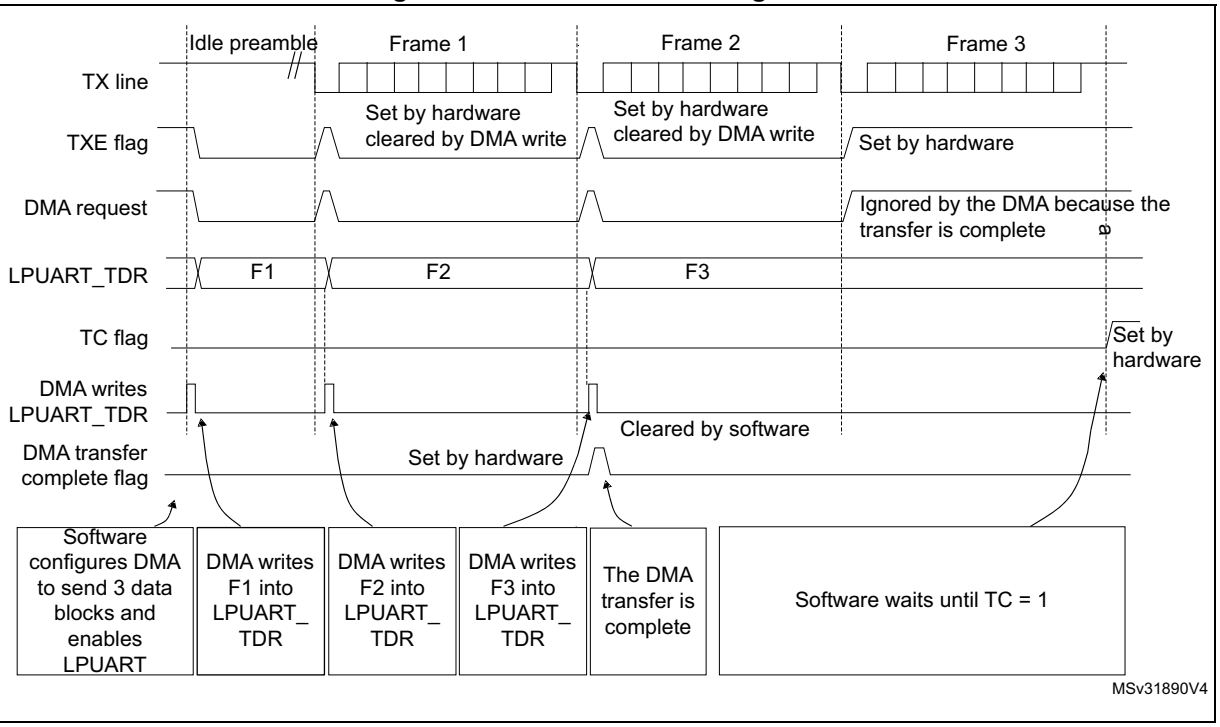
1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the LPUART_ISR register by setting the TCCF bit in the LPUART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (DMA transfer complete), the TC flag can be monitored to make sure that the LPUART communication has completed. This is required to avoid corrupting the last transmission before disabling the LPUART or entering low-power mode. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 958. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

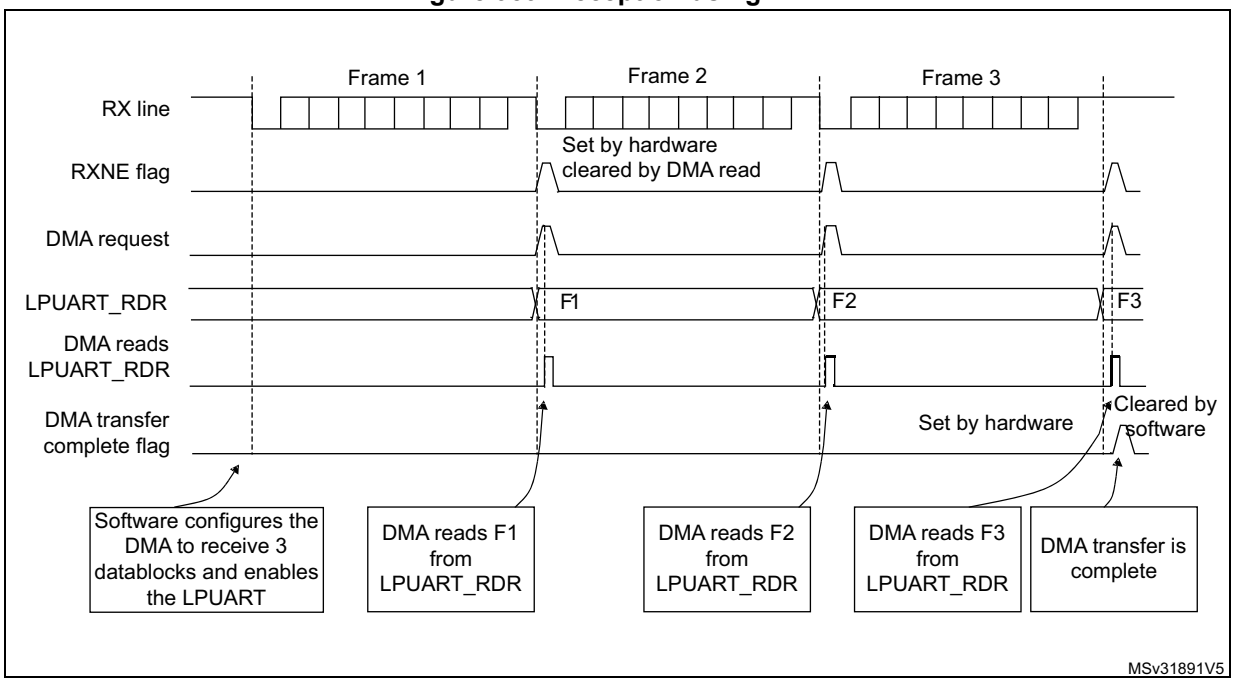
DMA mode can be enabled for reception by setting the DMAR bit in the LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to section *direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 959. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

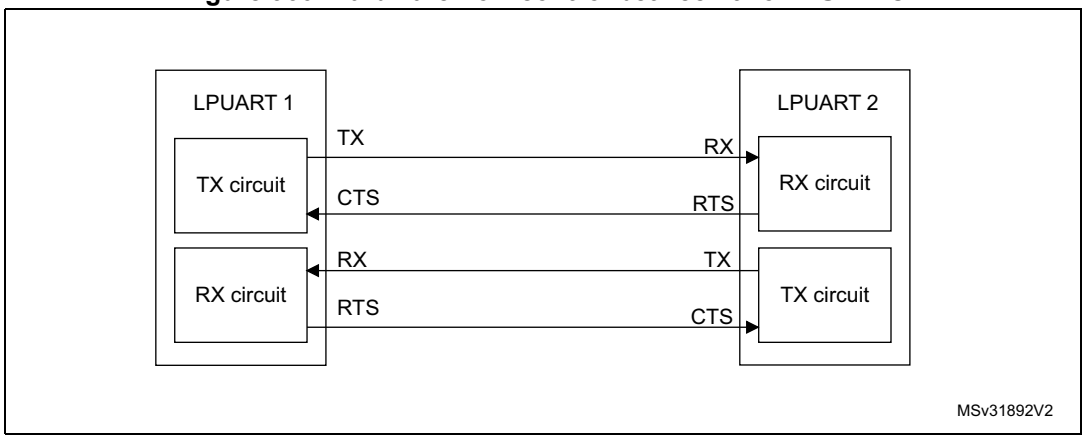
If any error occurs during a transaction In multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set.

For framing error, overrun error and noise flag, which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

47.4.14 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. *Figure 960* shows how to connect two devices in this mode.

Figure 960. Hardware flow control between two LPUARTs

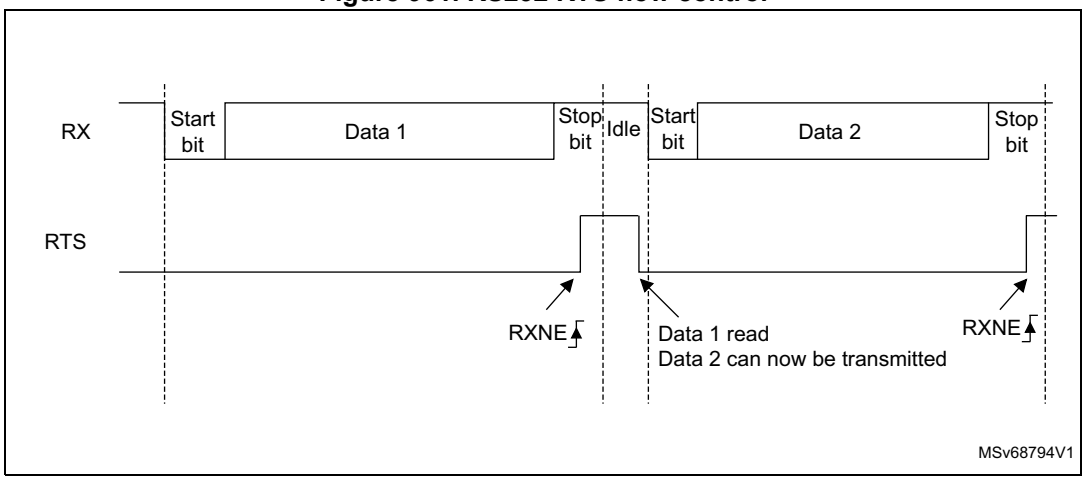


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 961* shows an example of communication with RTS flow control enabled.

Figure 961. RS232 RTS flow control



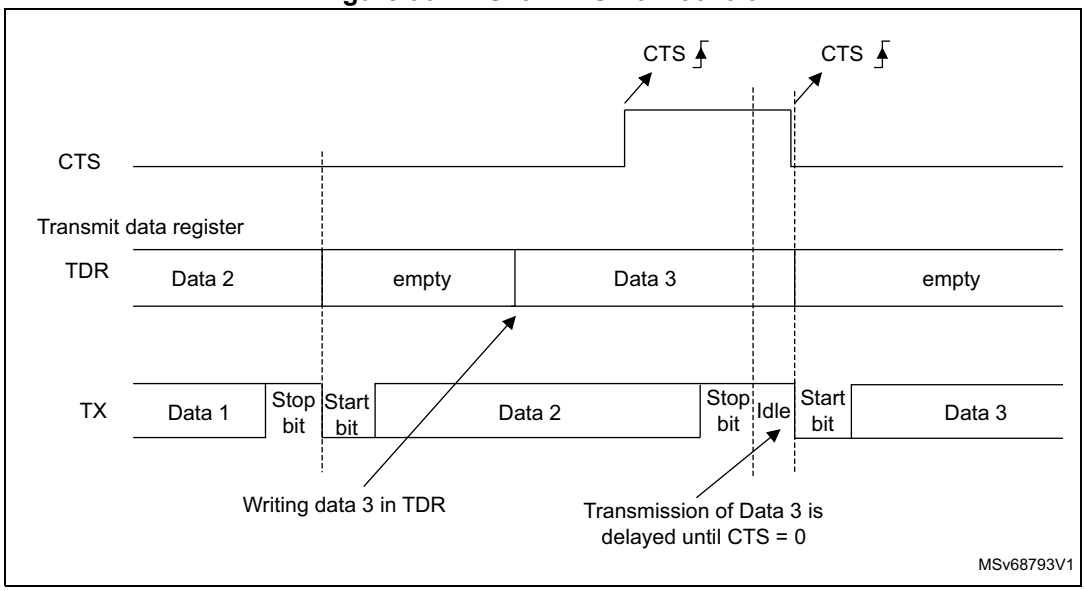
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART_CR3 register is set. *Figure 962* shows an example of communication with CTS flow control enabled.

Figure 962. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least 3 LPUART clock source periods before the end of the current character. In addition it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This enables activating the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the LPUART_CR1 control register. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the deactivation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the LPUART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART_CR3 control register.

The LPUART DEAT and DEDT are expressed in LPUART clock source ($f_{lpuart_ker_ck_pres}$) cycles:

- The driver enable assertion time equals
  - (1 + (DEAT x P)) x lpuart_ker_ck_pres, if P # 0
  - (1 + DEAT) x lpuart_ker_ck_pres, if P = 0
- The driver enable deassertion time equals
  - (1 + (DEDT x P)) x lpuart_ker_ck_pres, if P # 0
  - (1 + DEDT) x lpuart_ker_ck_pres, if P = 0

where P = BRR[20:11]

47.4.15 LPUART low-power management

The LPUART has advanced low-power mode functions that enable it to transfer properly data even when the lpuart_pclk clock is disabled.
The LPUART is able to wake up the MCU from low-power mode when the UESM bit is set.
When the lpuart_pclk is gated, the LPUART provides a wake-up interrupt (**lpuart_wkup**) if a specific action requiring the activation of the **lpuart_pclk** clock is needed:

- If FIFO mode is disabled

  luart_pclk clock has to be activated to empty the LPUART data register.

  In this case, the lpuart_wkup interrupt source is the RXNE set to 1. The RXNEIE bit must be set before entering low-power mode.

- If FIFO mode is enabled

  luart_pclk clock has to be activated
  - To fill the TXFIFO, or
  - To empty the RXFIFO

  In this case, the lpuart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the lpuart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many

LPUART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **lpuart_wkup** interrupt may be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and the **lpuart_wkup** interrupt is generated if the WUFIE bit is set.

Note: Before entering low-power mode, make sure that no LPUART transfer is ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or in an active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure that the LPUART is actually enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and reenabled upon exit from low-power mode.

When FIFO is enabled, the wake-up from low-power mode on address match is only possible when mute mode is enabled.

Using mute mode with low-power mode

If the LPUART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source from must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, mute mode is used with wake-up from low-power mode without any constraints (that is, the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wake-up from low-power mode when LPUART kernel clock lpuart_ker_ck is OFF in low-power mode

If during low-power mode, the lpuart_ker_ck clock is switched OFF, when a falling edge on the LPUART receive line is detected, the LPUART interface requests the lpuart_ker_ck clock to be switched ON thanks to the lpuart_ker_ck_req signal. The lpuart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, the lpuart_ker_ck is switched OFF again, the MCU is not woken up and stays in low-power mode and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.

Figure 963 shows the behavior when the wake-up event is verified.

Figure 963. Wake-up event verified (wake-up event = address match, FIFO disabled)

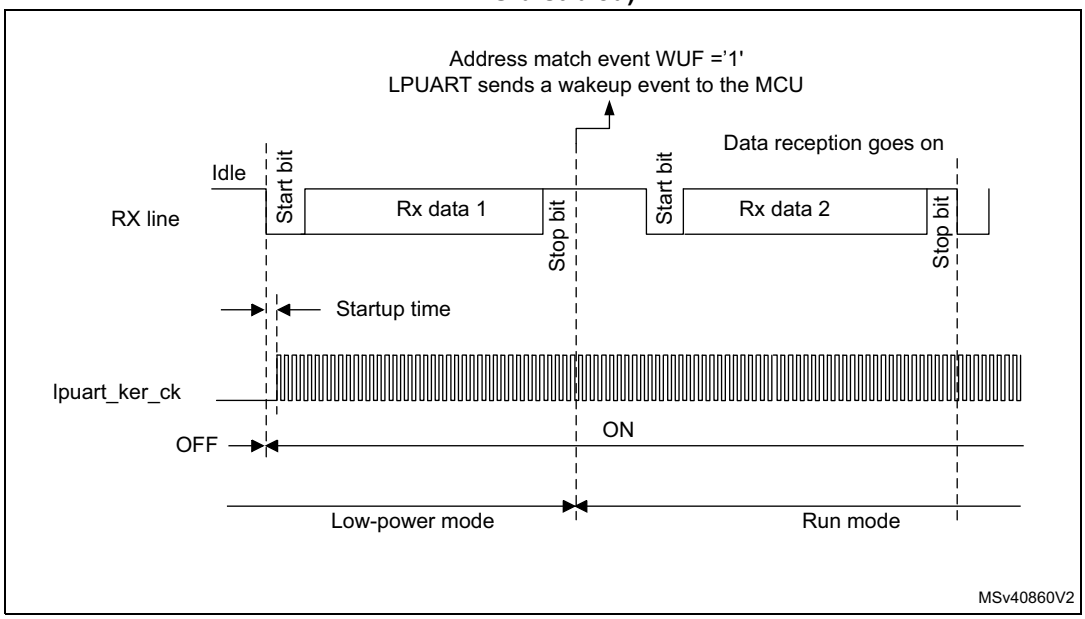
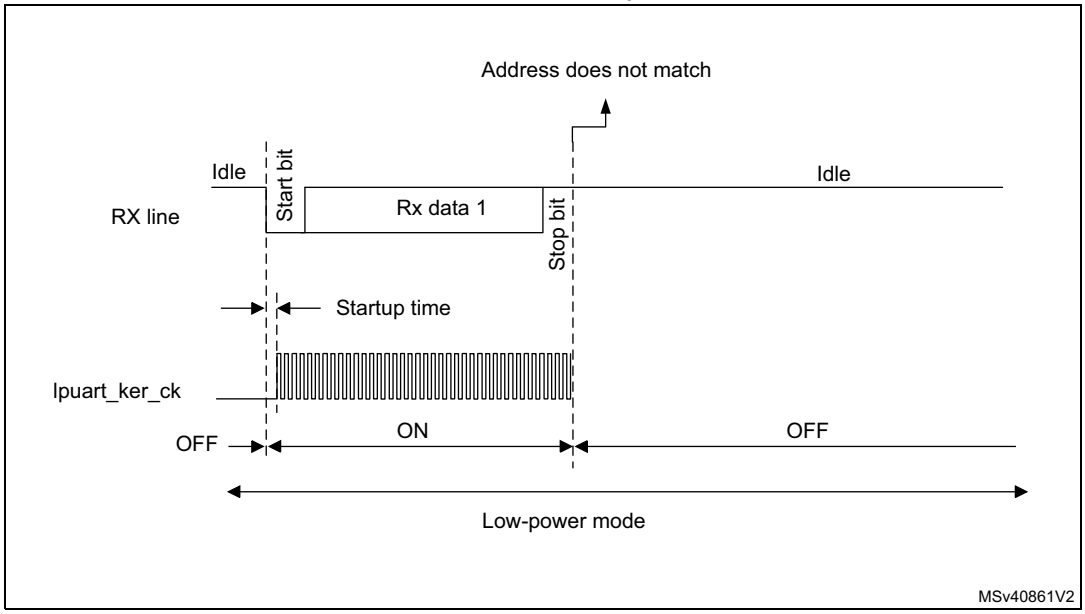


Figure 964 shows the behavior when the wake-up event is not verified.

Figure 964. Wake-up event not verified (wake-up event = address match, FIFO disabled)



Note: The above figures are valid when address match or any received frame is used as wake-up event. In the case the wake-up event is the start bit detection, the LPUART sends the wake-up event to the MCU at the end of the start bit.

Determining the maximum LPUART baud rate that enables to correctly wake up the MCU from low-power mode

The maximum baud rate that enables to correctly wake up the MCU from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see *Section 47.4.9: Tolerance of the LPUART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 548: Tolerance of the LPUART receiver*, the LPUART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < LPUART receiver tolerance

$D_{WUmax} = t_{WULPUART} / (11 \times T_{bit\ Min})$

$T_{bit\ Min} = t_{WULPUART} / (11 \times D_{WUmax})$

where $t_{WULPUART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the lpuart_ker_ck inaccuracy.

For example, if HSI is used as lpuart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WULPUART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = 3.41% – 1% = 2.41%

$T_{bit\ min}$ = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

47.5 LPUART in low-power modes

Table 550. Effect of low-power modes on the LPUART

Mode	Description
Sleep	No effect. LPUART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the LPUART registers is kept. The LPUART is able to wake up the microcontroller from Stop mode when the LPUART is clocked by an oscillator available in Stop mode.
Standby	The LPUART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 47.3: LPUART implementation* to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

47.6 LPUART interrupts

Refer to *Table 551* for a detailed description of all LPUART interrupt requests.

Table 551. LPUART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
LPUART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	No	No
	Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFIFO full		No	
	Transmit FIFO Empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission Complete	TC	TCIE	Write TDR or write 1 in TCCF		No	
	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	
	Receive FIFO Not Empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO Full	RXFF(2)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX FNEIE	Write 1 in ORECF		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		No	
	Overrun error in multibuffer communication.	ORE(3)		Write 1 in ORECF		No	
	Framing Error in multibuffer communication.	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Wake-up from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	

1. The LPUART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to *Section 47.3: LPUART implementation* for the list of supported Stop modes.
2. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART_RDR. In Stop mode, LPUART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
3. When OVRDIS = 0.

47.7 LPUART registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

47.7.1 LPUART control register 1 (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE:**RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when RXFF = 1 in the LPUART_ISR register

Bit 30 **TXFEIE:**TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when TXFE = 1 in the LPUART_ISR register

Bit 29 **FIFOEN:**FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = ‘10: 1 Start bit, 7 Data bits, n stop bits

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 46.5.21: RS232 hardware flow control and RS485 driver enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 47.4.14: RS232 hardware flow control and RS485 driver enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXFNF = 1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXFNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the LPUART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in low-power mode
When this bit is cleared, the LPUART cannot wake up the MCU from low-power mode.
When this bit is set, the LPUART can wake up the MCU from low-power mode.
This bit is set and cleared by software.
0: LPUART not able to wake up the MCU from low-power mode.
1: LPUART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: LPUART enable
When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.
0: LPUART prescaler and outputs disabled, low-power mode
1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.
The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

47.7.2 LPUART control register 1 [alternate] (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
		rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN:**FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = ‘10: 1 Start bit, 7 Data bits, n stop bits

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 46.5.21: RS232 hardware flow control and RS485 driver enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 47.4.14: RS232 hardware flow control and RS485 driver enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXE = 1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the LPUART acts as a transmitted, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to '1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in low-power mode

When this bit is cleared, the LPUART cannot wake up the MCU from low-power mode.

When this bit is set, the LPUART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

47.7.3 LPUART control register 2 (LPUART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
rw		rw	rw								rw				

Bits 31:24 **ADD[7:0]**: Address of the LPUART node

These bits give the address of the LPUART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter is equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter is equal to 1.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the reception is disabled (RE = 0) or when the LPUART is disabled (UE = 0).

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

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.
0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.
0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.
0: TX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)
1: TX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).
This enables the use of an external inverter on the TX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.
0: RX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0/mark)
1: RX pin signal values are inverted. ((V_{DD} = 0/mark, Gnd = 1 / idle).
This enables the use of an external inverter on the RX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.
0: TX/RX pins are used as defined in standard pinout
1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **STOP[1:0]**: Stop bits
These bits are used for programming the stop bits.
00: 1 stop bit
01: Reserved.
10: 2 stop bits
11: Reserved
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **ADDM7**:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the LPUART is disabled (UE = 0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

47.7.4 LPUART control register 3 (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	WUFIE	WUS1	WUS0	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth.
001:TXFIFO reaches 1/4 of its depth.
110:TXFIFO reaches 1/2 of its depth.
011:TXFIFO reaches 3/4 of its depth.
100:TXFIFO reaches 7/8 of its depth.
101:TXFIFO becomes empty.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG[2:0].

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth.
001:Receive FIFO reaches 1/4 of its depth.
110:Receive FIFO reaches 1/2 of its depth.
011:Receive FIFO reaches 3/4 of its depth.
100:Receive FIFO reaches 7/8 of its depth.
101:Receive FIFO becomes full.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG[2:0].

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: LPUART interrupt generated whenever WUF = 1 in the LPUART_ISR register

Note: WUFIE must be set before entering in low-power mode.
If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 47.3: LPUART implementation.*

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 47.3: LPUART implementation.*

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.

This bit can only be written when the LPUART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.

This bit can only be written when the LPUART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register.

47.7.5 LPUART control register 3 [alternate] (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS1	WUS0	Res.	Res.	Res.	Res.
									rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

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

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: LPUART interrupt generated whenever WUF = 1 in the LPUART_ISR register

Note: WUFIE must be set before entering in low-power mode.
If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 47.3: LPUART implementation.*

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 47.3: LPUART implementation.*

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA Disable on reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: This control bit enables *checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF = 1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.
This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 **RTSE**: RTS enable
0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register.

47.7.6 LPUART baud rate register (LPUART_BRR)

This register can only be written when the LPUART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **BRR[19:0]:** LPUART baud rate division (LPUARTDIV)

Note: It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

Provided that LPUART_BRR must be ≥ 0x300 and LPUART_BRR is 20 bits, a care must be taken when generating high baud rates using high lpuart_ker_ck_pres values. lpuart_ker_ck_pres must be in the range [3 x baud rate..4096 x baud rate].

47.7.7 LPUART request register (LPUART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

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

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

Bit 4 **TXFRQ**: Transmit data flush request

This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This enables discarding the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the LPUART in Mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 Reserved, must be kept at reset value.

47.7.8 LPUART interrupt and status register (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0080 00C0

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	Res.	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
				r	r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of LPUART_CR3 register.An interrupt is generated if the TXFTIE bit = 1 (bit 31) in the LPUART_CR3 register.

0: TXFIFO does not reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield of the LPUART_CR3 register, that is, the Receive FIFO contains RXFTCFG[2:0] data. An interrupt is generated if the RXFTIE bit = 1 (bit 28) in the LPUART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the LPUART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th data.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the LPUART_RDR register.

An interrupt is generated if the RXFFIE bit = 1 in the LPUART_CR1 register.

0: RXFIFO is not Full.

1: RXFIFO is Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART_RQR register.

An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the LPUART_CR1 register.

0: TXFIFO is not empty.

1: TXFIFO is empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.
An interrupt is generated if WUFIE = 1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 47.3: LPUART implementation.*

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.
When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.
0: Receiver in active mode
1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.
0: No break character transmitted
1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.
An interrupt is generated if CMIE = 1in the LPUART_CR1 register.
0: No Character match detected
1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).
0: LPUART is idle (no reception)
1: reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.
0: CTS line set
1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.

The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

An interrupt is generated if the TXFNFIE bit = 1 in the LPUART_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data has completed and the TXFE bit is set.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the LPUART_ICR register, or by a write to the LPUART_TDR register.

0: Transmission has not completed

1: Transmission has completed

Note: If the TE bit is reset and no transmission is ongoing, the TC bit is immediately set.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART_RDR register. Every read of the LPUART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

This error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.

When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the LPUART_RDR.

47.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

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

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.

An interrupt is generated if WUFIE = 1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 47.3: LPUART implementation.*

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE = 1in the LPUART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: Reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the LPUART_TDR register has been transferred into the shift register. It is cleared by a write to the LPUART_TDR register.

An interrupt is generated if the TXEIE bit = 1 in the LPUART_CR1 register.

0: Data register full

1: Data register empty

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the LPUART_ICR register or by writing to the LPUART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the LPUART_RDR shift register has been transferred to the LPUART_RDR register. It is cleared by a read to the LPUART_RDR register. The

RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXNE = 1 (RXFF = 1 in case FIFO mode is enabled). It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXNEIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE/RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the LPUART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.
An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.
0: No parity error
1: Parity error

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

47.7.10 LPUART interrupt flag clear register (LPUART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
						w			w		w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the LPUART_ISR register.

Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value.* Refer to *Section 47.3: LPUART implementation.*

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the LPUART_ISR register.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the LPUART_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the LPUART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the LPUART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the LPUART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the LPUART_ISR register.

47.7.11 LPUART receive data register (LPUART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see *Section 47.4.1: LPUART block diagram*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

47.7.12 LPUART transmit data register (LPUART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

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

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

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see *Section 47.4.1: LPUART block diagram*).

When transmitting with the parity enabled (PCE bit set to 1 in the LPUART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

47.7.13 LPUART prescaler register (LPUART_PRESC)

This register can only be written when the LPUART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

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

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

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The LPUART input clock can be divided by a prescaler:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Others: Reserved, must not be used.

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011, that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 47.3: LPUART implementation.

47.7.14 LPUART register map

Table 552. LPUART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	LPUART_CR1FIFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]						DEDT[4:0]				Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	LPUART_CR1FIFO mode disabled	Res.	Res.	FIFOEN	M1	Res.	Res.	DEAT[4:0]						DEDT[4:0]				Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	LPUART_CR2	ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	Res.	STOP [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0		0	0								0				
0x08	LPUART_CR3 FIFO mode enabled	TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	WUFIE	WUS 1	WUS 0	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	Res.	EIE
	Reset value	0	0	0	0	0	0	0		0	0	0	0					0	0	0	0		0	0	0	0	0			0			0
0x08	LPUART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS 1	WUS 0	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	Res.	EIE
	Reset value										0	0	0					0	0	0	0		0	0	0	0	0			0				0
0x0C	LPUART_BRR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	BRR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10-0x14	Reserved																																
0x18	LPUART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
	Reset value																												0	0	0	0	
0x1C	LPUART_ISRFIFO mode enabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFF	TXFF	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value								0	1	0	0	0	0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x1C	LPUART_ISRFIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value										0	0	0	0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
	Reset value												0			0								0			0		0	0	0	0	0
0x24	LPUART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0

Table 552. LPUART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	LPUART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	LPUART_ PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

48 Serial peripheral interface (SPI)

48.1 SPI introduction

The serial peripheral interface (SPI) can be used to communicate with external devices while using the specific synchronous protocol. The SPI protocol supports half-duplex, full-duplex, and simplex synchronous, serial communication with external devices. The interface can be configured as master or slave and can operate in multislave or multimaster configurations. The device configured as master provides a communication clock (SCK) to the slave device. The slave select (NSS) and ready (RDY) signals can be applied optionally just to set up communication with a concrete slave and to ensure it handles the data flow properly. The Motorola data format is used by default, but some other specific modes are supported as well.

48.2 SPI main features

- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- From 4-bit up to 32-bit data size selection
- Multimaster or multislave mode capability
- Dual clock domain, the peripheral kernel clock is independent from the APB bus clock
- Baud rate prescaler up to kernel frequency/2 or bypass from RCC in master mode
- Protection of configuration and setting
- Slave select hardware or software management for both master and slave
- Adjustable minimum delays between data and between NSS and data flow
- Configurable NSS signal polarity and timing, MISO x MOSI swap capability
- Programmable clock polarity and phase with optional sampling delay when reading master data
- Programmable data order with MSB-first or LSB-first shifting
- Programmable number of data within a transfer to control NSS and CRC
- Dedicated transmission and reception flags with interrupt capability
- SPI Motorola and TI format support
- Hardware CRC can verify the integrity of the communication at the end of a transfer by:
  - Adding CRC value in Tx mode
  - Automatic CRC error checking for Rx mode

- Error detection with interrupt capability in case of data overrun, CRC error, data underrun, the mode fault, and frame error, depending on the operating mode
- Two 8-bit width embedded Rx and Tx FIFOs (FIFO size depends on instances)
- Configurable FIFO thresholds (to handle the data packets)
- Capability to handle data streams by system DMA controller
- Configurable behavior at slave underrun condition (support of cascaded circular buffers)
- Optional status pin RDY signalizing that the slave device is ready to handle the data flow

48.3 SPI implementation

The table below describes the SPI implementation.

Table 553. SPI features

SPI feature	SPI1, SPI2, SPI3 (full-featured instances)	SPI4, SPI5, SPI6 (full-featured instances)
Data and CRC size	Configurable from 4 to 32 bits	Configurable from 4 to 16 bits
CRC computation	CRC polynomial length configurable from 5 to 33 bits	CRC polynomial length configurable from 5 to 17 bits
FIFO size	16x8 bits	8x8 bits
Number of data control (TSIZE)	Up to 65535	Up to 65535
I2S feature	Yes	No
Wake-up in slave mode from Stop SVOS high mode	Yes	Yes

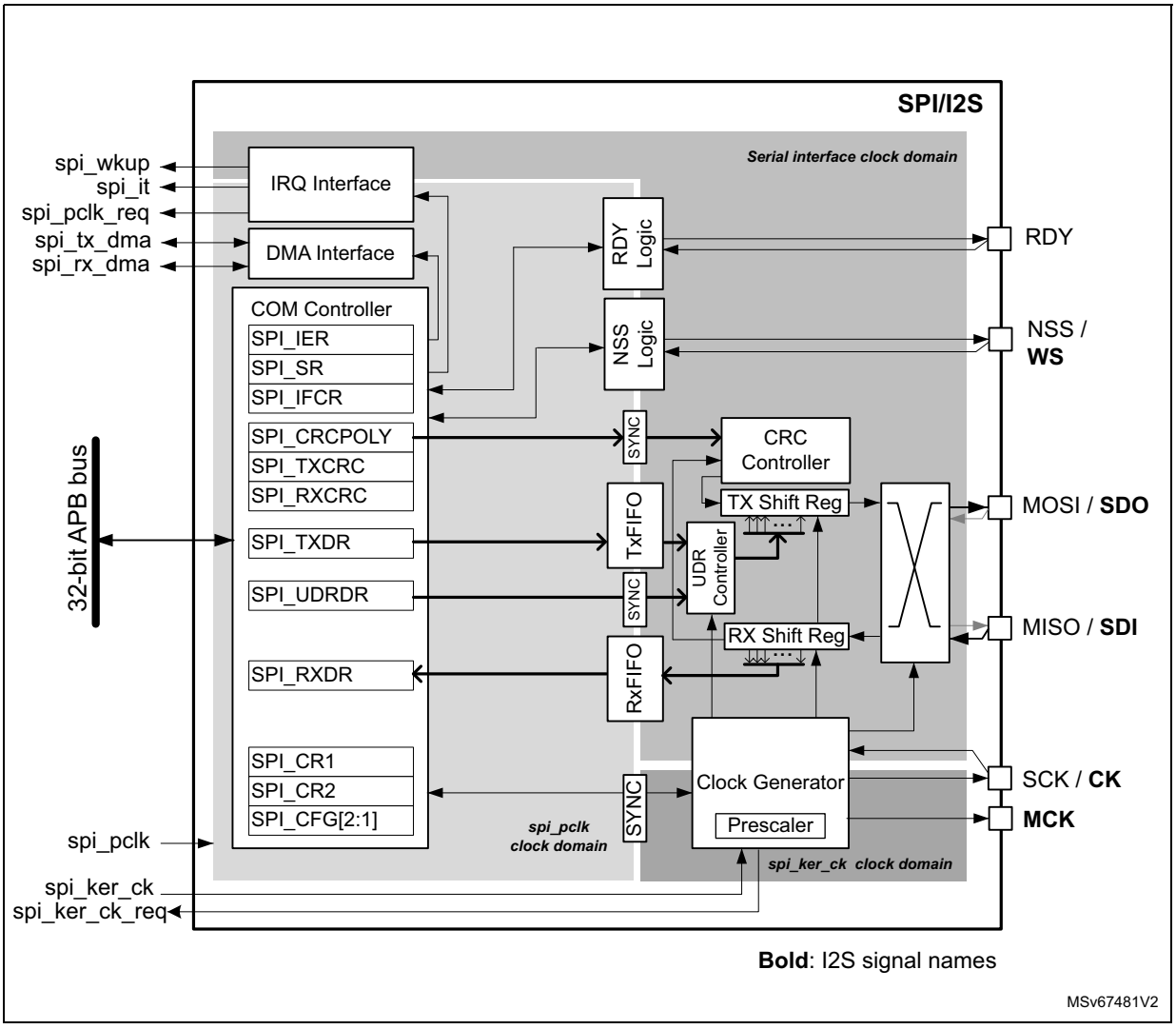
Note: For detailed information about instance capabilities to exit from Stop and Standby modes, refer to Table 557: SPI wake-up and interrupt requests.

48.4 SPI functional description

48.4.1 SPI block diagram

The SPI enables synchronous, serial communications between the MCU and external devices. The application software can manage the communication by polling the status flag or using a dedicated SPI interrupt. The main SPI elements and their interactions are shown in *Figure 965*.

Figure 965. SPI/I2S block diagram



The simplified scheme of *Figure 965* shows three fully independent clock domains:

- The **spi_pclk** clock domain
- The **spi_ker_ck** kernel clock domain
- The serial interface clock domain

All the control and status signals between these domains are strictly synchronized. There is no specific constraint concerning the frequency ratio between these clock signals. The user

has to consider a ratio compatible with the data flow speed to avoid data underrun or overrun events.

The **spi_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the SPI registers are required.

The SPI working in slave mode handles a data flow using the serial interface clock derived from the external SCK signal provided by the external master SPI device. That is why the SPI slave is able to receive and send data even when the **spi_pclk** and **spi_ker_ck** clock signals are inactive. As a consequence, a specific slave logic working within the serial interface clock domain needs some additional traffic to be set up correctly (for example when underrun or overrun is evaluated, see *Section 48.5.2* for details). This cannot be done when the bus becomes idle. In some specific cases, the slave even requires the clock generator working (see *Section 48.5.1*).

When the SPI works as a master, the RCC must provide the **spi_ker_ck** kernel clock to the peripheral during communication to feed the serial interface clock via the clock generator where it can be divided by prescaler or bypassed. The signal is then provided to the slaves via the SCK pin and internally to the serial interface domain of the master.

48.4.2 SPI pins and internal signals

Up to five I/O pins are dedicated to SPI communication with external devices.

- **MISO:** master in / slave out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** master out / slave in data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** serial clock output pin for SPI masters and input pin for SPI slaves.
- **NSS:** slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - Select an individual slave device for communication
  - Synchronize the data frame, or
  - Detect a conflict between multiple masters

  See *Section 48.4.7* for details.
- **RDY:** optional status pin signaling slave FIFO occupancies and so the slave availability to continue the transfer without any risk of data flow corruption. It can be checked by the master to control the temporal suspension of the ongoing communication.

All these pins (except RDY) are shared in the I2S mode. This mode features an additional I2S specific MCK signal. For more details about I2S signals, see *Section 48.9.2*.

The SPI bus enables the communication between one master device and one or more slave devices. The bus consists of at least two wires: one for the clock signal and the other for synchronous data transfer. Other signals are optional and can be added depending on the data exchange between SPI nodes and their communication control management.

Refer to *Table 554* and *Table 555* for the list of SPI input / output pins and internal signals.

Table 554. SPI/I2S input/output pins(1)

Pin name	I/O type	Description
MISO/SDI(2)	Input/output	Master data input / slave data output
MOSI/SDO(2)	Input/output	Master data output / slave data input
SCK/CK	Input/output	Master clock output / slave clock input
NSS/WS	Input/output	Master output / slave selection input
RDY	Input/output	SPI master input / slave FIFOs status occupancy output
MCK	Output	I2S master frequency output

1. Refer to the section *Section 48.9.2: Pin sharing with SPI function* for details.
2. Functionality of MOSI/SDO and MISO/SDI pins can be swapped. Their directions may vary in SPI bidirectional half-duplex mode.

Description of SPI input/output signals

Table 555. SPI internal input/output signals

Signal name	Signal type	Description
spi_pclk	Input	SPI clock signal feeds the peripheral bus interface
spi_ker_ck	Input	SPI kernel clock
spi_ker_ck_req	Output	SPI kernel clock request
spi_pclk_req	Output	SPI clock request
spi_wkup	Output	SPI provides a wake-up interrupt
spi_it	Output	SPI global interrupt
spi_tx_dma	Input/output	SPI transmit DMA request
spi_rx_dma	Input/output	SPI receive DMA request

48.4.3 SPI communication general aspects

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use two or three wires (with software slave select management) or three or four wires (with hardware slave select management). The communication is always initiated and controlled by the master. The master provides a clock signal on the SCK line and selects or synchronizes slaves for communication by NSS line when it is managed by hardware.

The data between the master and the slave flow synchronously on the MOSI and/or MISO lines.

48.4.4 Communications between one master and one slave

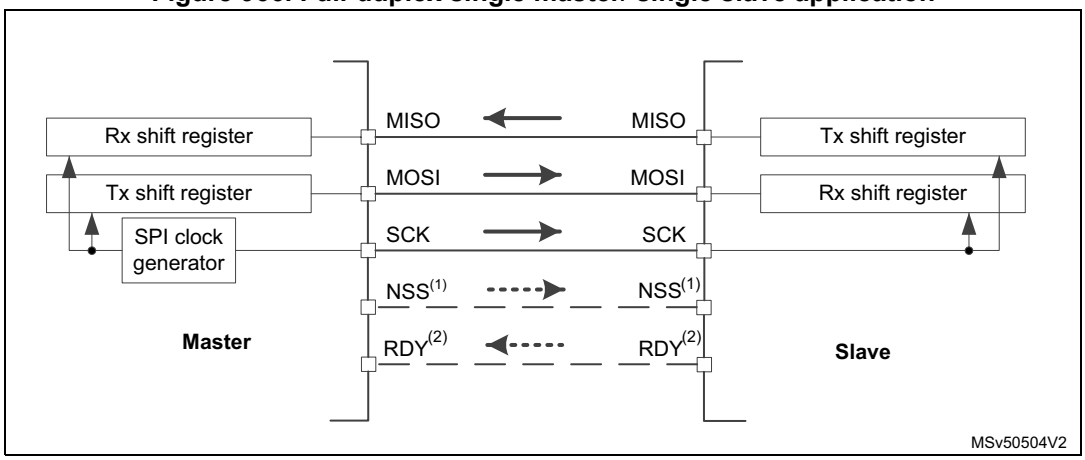
The communication flow can use one of three possible modes: the full-duplex (three wires) mode, half-duplex (two wires) mode, or the simplex (two wires) mode. The NSS signal is optional in single master-slave configuration and is often not connected between the two communication nodes. Nevertheless, the NSS signal can be helpful in this configuration to synchronize the data flow and it is used by default for some specific SPI modes (for example the TI mode).

The next optional RDY signal can help to ensure the correct management of all the transferred data at slave side.

Full-duplex communication

By default, the SPI is configured for full-duplex communication (bits COMM[1:0] = 00 in the SPI_CFG2 register). In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During the SPI communication, the data are shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line simultaneously. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 966. Full-duplex single master/ single slave application

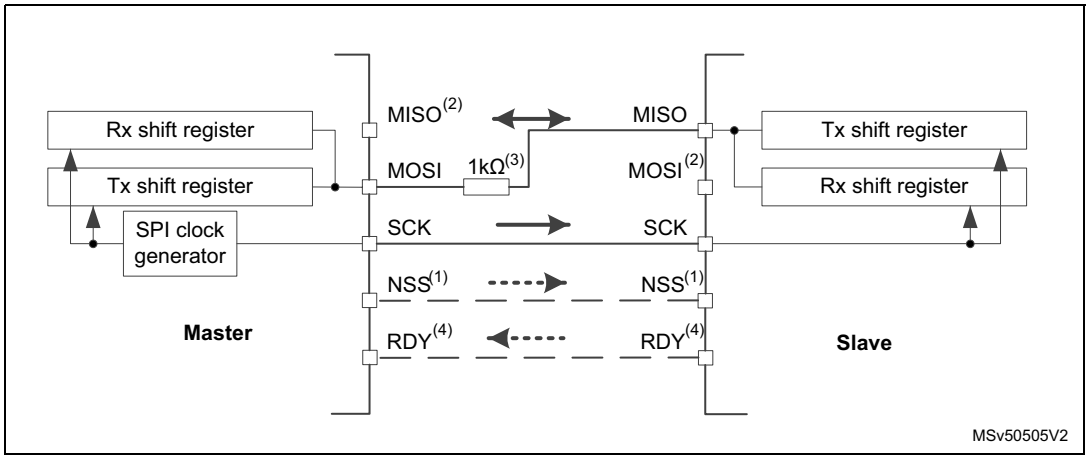


1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 48.4.7*).
2. The RDY signal provided by the slave can be read by the master optionally.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting COMM[1:0] = 11 in the SPI_CFG2 register. In this configuration, one single cross-connection line is used to link the shift registers of the master and slave together. During this communication, the data are synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the HDDIR bit in their SPI_CR1 registers. Note that the SPI must be disabled when changing the direction of the communication. In this configuration, the MISO pin at master and the MOSI pin at slave are free for other application uses and act as GPIOs.

Figure 967. Half-duplex single master/ single slave application



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 48.4.7*).
2. In this configuration, the MISO pin at master and MOSI pin at slave can be used as GPIOs
3. A critical situation can happen when the communication direction is not changed synchronously between two nodes working in bidirectional mode. The new transmitter accesses the common data line while the former transmitter still keeps an opposite value on the line (the value depends on the SPI configuration and communicated data). The nodes can conflict temporarily with opposite output levels on the line until the former transmitter changes its data direction setting. It is suggested to insert a serial resistance between MISO and MOSI pins in this mode to protect the conflicting outputs and limit the current flow between them.
4. The RDY signal provided by the slave can be read by the master optionally.

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the COMM[1:0] field in the SPI_CFG2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO or MOSI pin pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode: COMM[1:0] = 01**

  The master in transmit-only mode generates the clock as long as there are data available in the TxFIFO and the master transfer is ongoing.

  The slave in transmit-only mode sends data as long as it receives a clock on the SCK pin and the NSS pin (or software managed internal signal) is active (see *Section 48.4.7*).

- **Receive-only mode: COMM[1:0] = 10**

  In master mode, the MOSI output is disabled and can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled and the CSTART bit in the SPI_CR1 register is set. The clock is stopped either by software explicitly requesting this by setting the CSUSP bit in the SPI_CR1 register or automatically when the RxFIFO is full, when the MASRX bit in the SPI_CR1 is set.

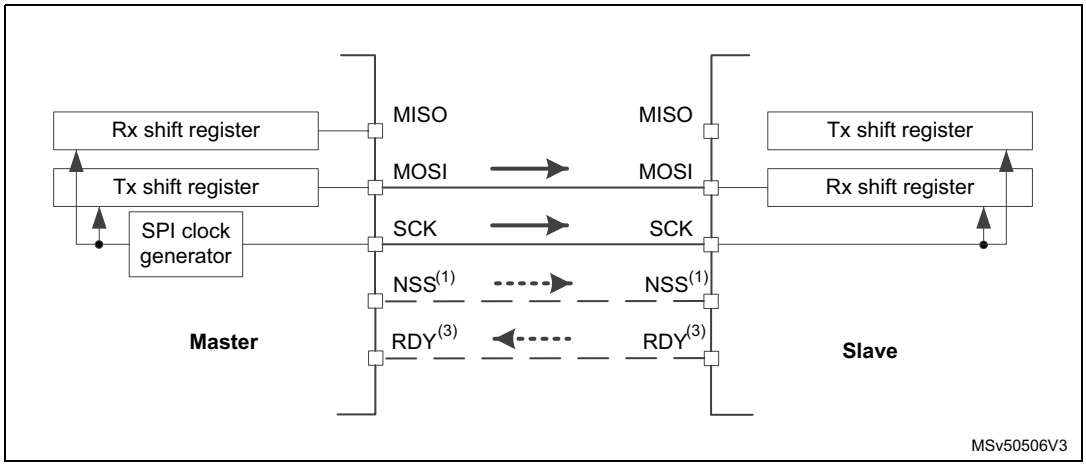
  In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see *Section 48.4.7*).

Note: In whatever master and slave modes, the data pin dedicated for transmission can be replaced by the data pin dedicated for reception and vice versa by changing the IOSWP bit value in the SPI_CFG2 register (this bit can only be modified when the SPI is disabled).

Any simplex communication can be replaced by a variant of the half-duplex communication with a constant setting of the transfer direction (bidirectional mode is enabled, while the

HDDIR bit is never changed) or by full-duplex control when unused data line and corresponding data flow is ignored.

Figure 968. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode)



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 48.4.7*).
2. In this configuration, both the MISO pins can be used as GPIOs.
3. The RDY signal provided by the slave can be read by the master optionally.

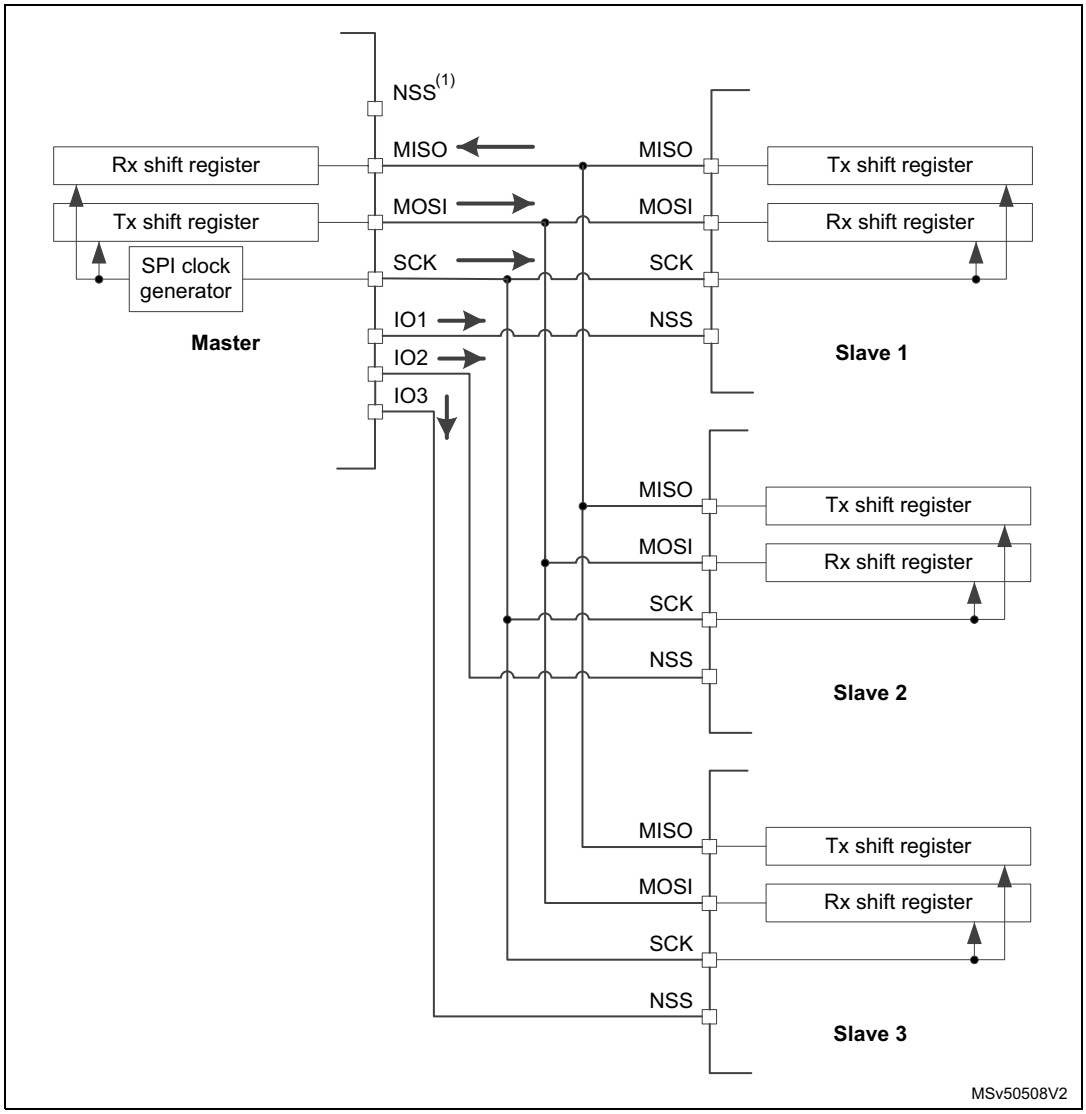
48.4.5 Standard multislave communication

In a configuration with two or more independent slaves, the master uses a star topology with dedicated GPIO pins to manage the chip select lines for each slave separately (see *Figure 969.*).

The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input (only one slave can control data on a common MISO line at a given time).

When this is done, a communication between the master and the selected slave is established. In addition to the simplicity, the advantage of this topology is that a specific SPI configuration can be applied for each slave as all the communication sessions are performed separately just within a single master-slave pair. Optionally, when there is no need to read any information from slaves, the master can transmit the same information to the multiple slaves.

Figure 969. Master and three independent slaves connected in star topology



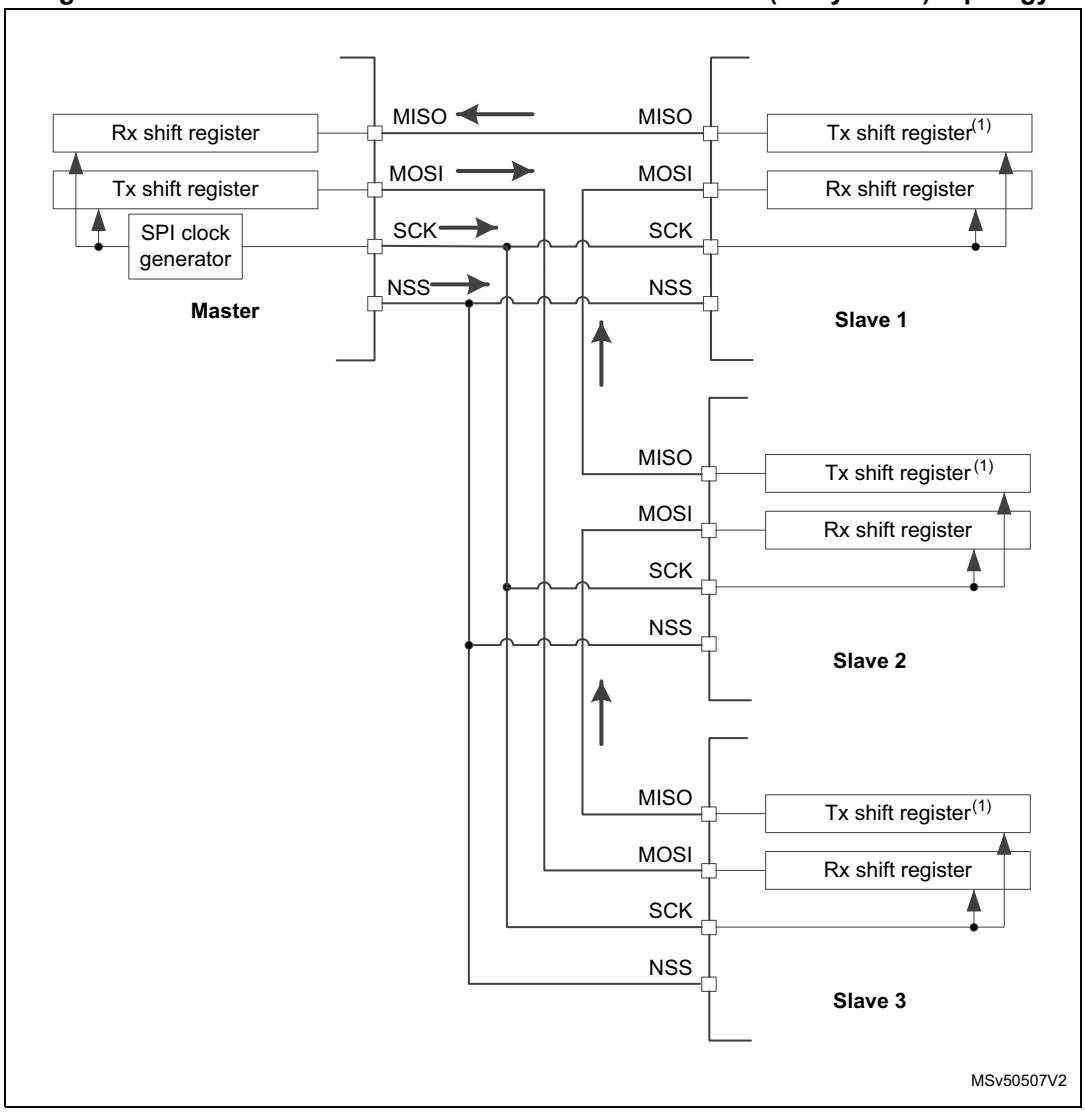
1. The master single NSS pin hardware output functionality cannot support this topology (to be replaced by a set of GPIOs under software control). The NSS pin is free for other application uses (such as GPIO or other alternate functions). Refer to *Section 48.4.7* for details.
2. If the application cannot ensure that no more than a single NSS active signal is provided by the master at a given time, it is better to configure the MISO pins in an open-drain configuration with an external pull-up on the MISO line to prevent conflicts between the interconnected outputs of the slaves. Else, a push-pull configuration can be applied without an extra resistor (see I/O alternate function input/output (GPIO) section).
3. The RDY signals can be read by the master from the slaves optionally.

The master can handle the SPI communication with all the slaves at the same time when a circular topology is applied (see *Figure 970*). All the slaves behave like simple shift registers applied in serial chain under control of common slave select (NSS) and clock (SCK) signals. All the information is shifted simultaneously around the circle while returning back to the master. Sessions have fixed the length where the number of data frames transferred by the master is equal to the number of slaves.

Then when a first data frame is transferred in the chain, the master just sends the information dedicated for the last slave node in the chain, via the first slave node input, while the first information received by the master comes from the last node output at this time.

Correspondingly, the last transferred data finishing the session is dedicated for the first slave node while its first outgoing data just reaches the master input, after its circling around the chain passing through all the other slaves during the session.
The data format configuration and clock setting must be the same for all the nodes in the chain in this topology. As the receive and transmit shift registers are separated internally, a trick with intentional underrun must be applied to the TxFIFO slaves when the information is transferred between the receiver and the transmitter by hardware.
In this case, the transmission underrun feature is configured in a mode repeating the last received data frame (UDRCFG=1). A session can start optionally with a single data pattern written into the TxFIFO by each slave (usually slave status information is applied) before the session starts. In this case, the underrun happens in fact after this first data frame is transferred. To be able to clear the internal underrun condition immediately and restart the session by the TxFIFO content again, the user must disable and enable the SPI between the sessions and must fill the TxFIFO by a new single data pattern (to overcome the propagating delay of the clearing raised in case the underrun is cleared in a standard way by the UDRC bit).

Figure 970. Master and three slaves connected in circular (daisy chain) topology



1. The underrun feature is used by the slaves in this configuration when the slaves are able to transmit data received previously into the Rx shift register once their TxFIFOs become empty.
2. The RDY signals can be read by the master optionally, either separately or configured as open drain outputs (while RDIOP = 0) and connected with a pull-up resistor as a common chain ready status overdriven by the slowest device.

48.4.6 Multimaster communication

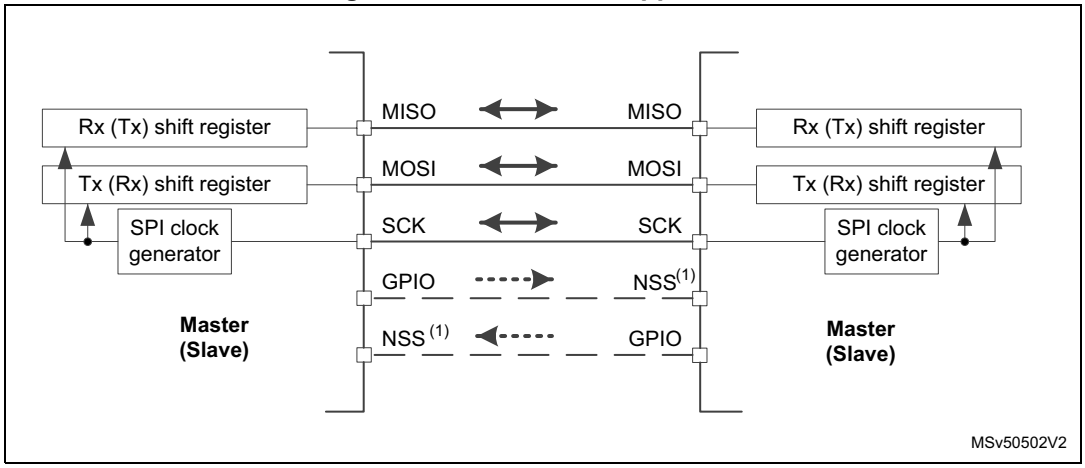
Unless the SPI bus is not designed primarily for a multimaster capability, the user can use a built-in feature that detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, the NSS pin is configured in hardware input mode. The connection of more than two SPI nodes working in this mode is impossible, as only one node can apply its output on a common data line at a given time.

When the nodes are not active, both stay in slave mode by default. Once a node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via the dedicated GPIO pin. After the session is

complete, the active slave select signal is released and the node mastering the bus temporarily returns back to passive slave mode waiting for the next session to start.

If both nodes raise their mastering request at the same time, a bus conflict event appears (see mode fault MODF event). The user can apply some simple arbitration process (for example postpone the next attempt by different predefined timeouts applied to both nodes).

Figure 971. Multimaster application



1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control since the passive node is configured as a slave.
2. The RDY signal is not used in this communication.

48.4.7 Slave select (NSS pin) management

In slave mode, the NSS works as a standard ‘chip select’ input and lets the slave communicate with the master. In master mode, the NSS can be used either as an output or an input. As an input it can prevent a multimaster bus collision, and as an output it can drive a slave select signal of a single slave. The NSS signal can be managed internally (software management of the NSS input) or externally when both the NSS input and output are associated with the NSS pin (hardware slave select management). The user can configure which level of this input/output external signal (present on the NSS pin) is considered as active by the SSIOP bit setting. NSS level is considered as active if it is equal to SSIOP.

The hardware or software slave select management can be set using the SSM bit in the SPI_CFG2 register:

- **Software slave select management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in the SPI_CR1 register. The external NSS pin is free for other application uses (such as GPIO or other alternate functions).
- **Hardware slave select management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in the SPI_CFG2 register).
  - **NSS output enable (SSOE = 1)**: this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The functionality is tied to CSTART and EOT control. As a consequence, the master must apply the proper TSIZE > 0 setting to control the NSS output correctly. Even if SPI AF is not applied at the NSS pin (it can be used as a standard GPIO then), keep anyway SSOE = 1 to ensure the default NSS input level and prevent any mode fault

evaluation at the input of the master NSS internal logic applicable at a multimaster topology exclusively.

a) When SSOM = 0 and SP = 000, the NSS signal is driven to the active level as soon as the master transfer starts (CSTART = 1) and it is kept active until its EOT flag is set or the transmission is suspended.

b) When SP = 001, a pulse is generated as defined by the TI mode.

c) When SSOM = 1, SP = 000 and MIDI > 1 the NSS is pulsed inactive between data frames, and kept inactive for a number of SPI clock periods defined by the MIDI value decremented by one (from 1 to 14).

d) NSS input is forced to nonactive state internally at master to prevent any mode fault.

– **NSS output disable (SSM = 0, SSOE = 0)**:

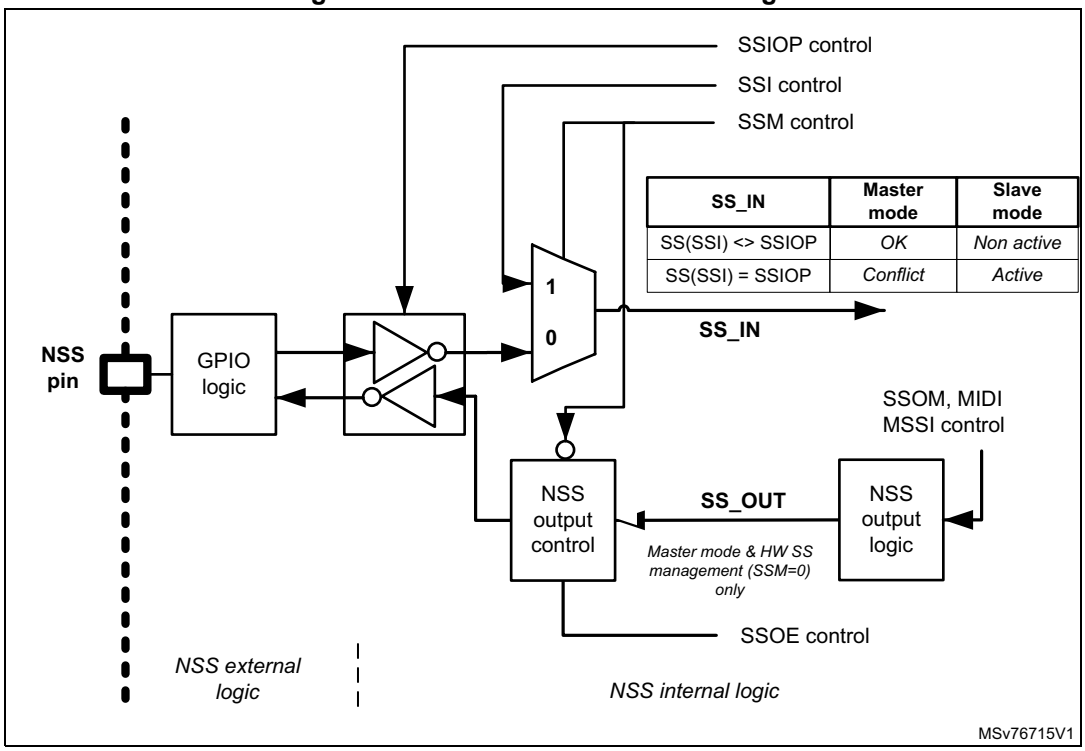
a) If the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled into an active level in this mode, the SPI enters master mode fault state and the SPI device is automatically reconfigured in slave mode (MASTER = 0).

b) In slave mode, the NSS pin works as a standard ‘chip select’ input and the slave is selected while the NSS line is at its active level.

Note: The purpose of automatic switching into slave mode when a mode fault occurs is to avoid the possible conflicts on data and clock lines. As the SPE is automatically reset in this condition, both Rx and Tx FIFOs are flushed and current data is lost.

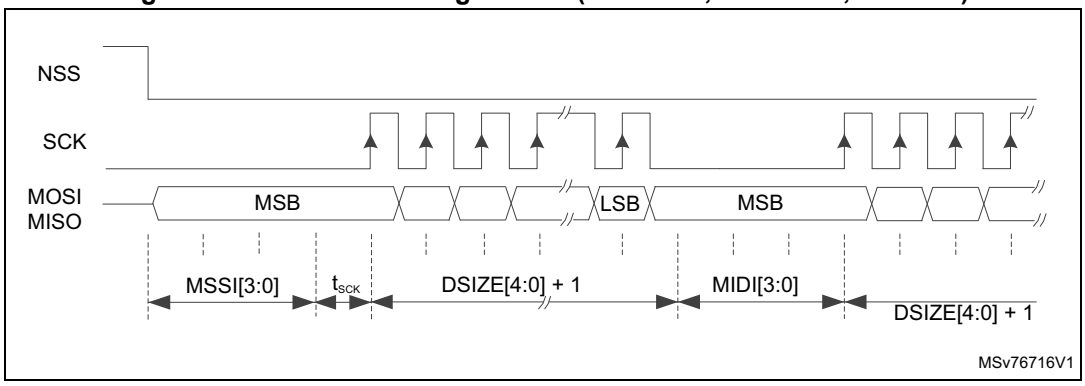
When the SPI slave is enabled in the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (transition from nonactive to active level) just synchronizing the slave with the master. This is because the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection in the slave select software management. Then the SSI bit must be changed when there is no traffic on the bus and the SCK signal is in idle state level between transfers exclusively in this case.

Figure 972. Scheme of NSS control logic



When the hardware output NSS control is applied (SSM = 0, SSOE = 1), by configuration of the MIDI[3:0] and MSSI[3:0] bitfields, the user can control the timing of the NSS signal between data frames and can insert an extra delay at the beginning of every transfer (to separate the NSS and clock starts). This can be useful when the slave needs to slow down the flow to obtain sufficient room for correct data handling (see *Figure 973*).

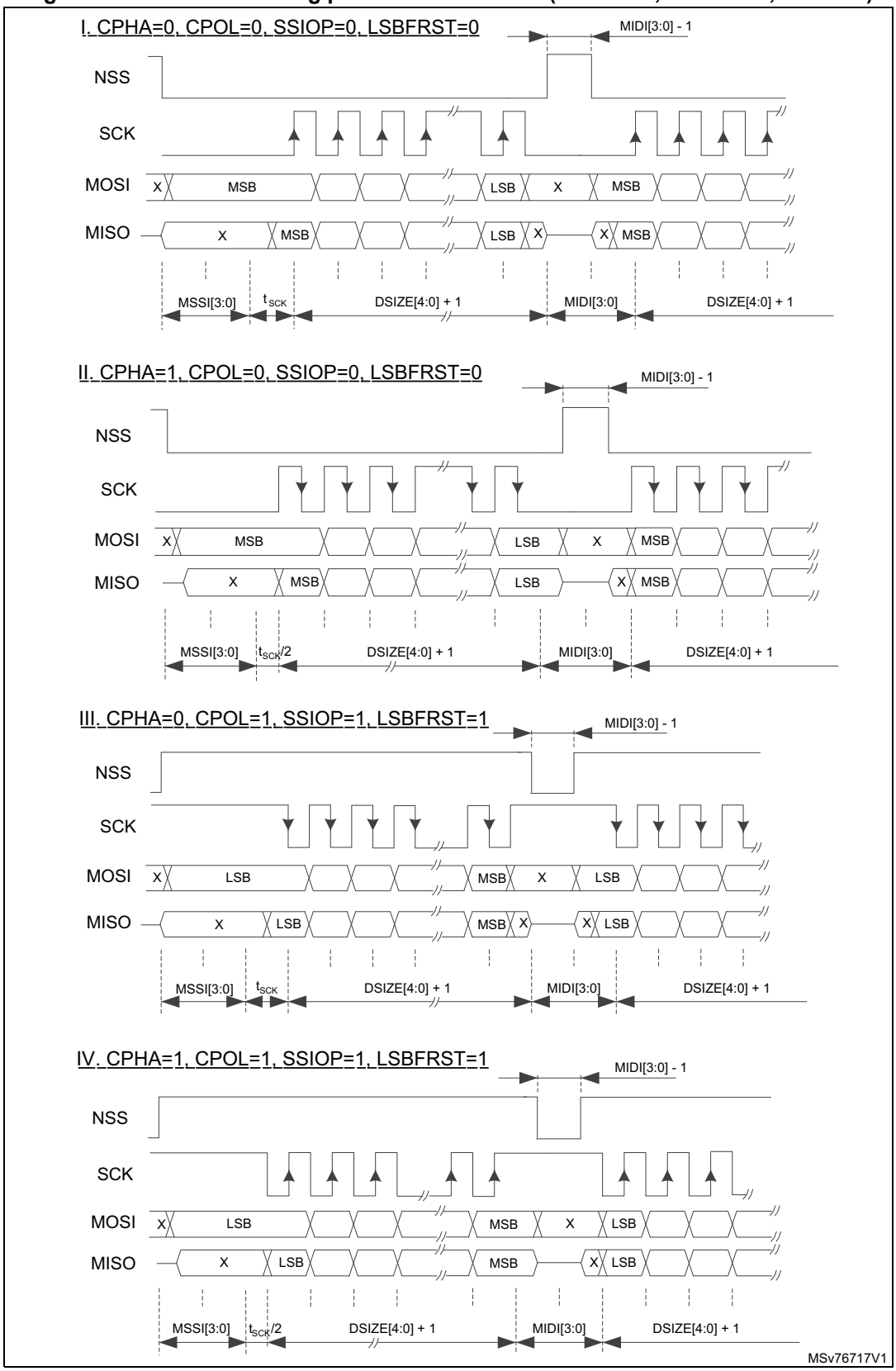
Figure 973. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0)



1. MSSI[3:0] = 0011, MIDI[3:0] = 0011 (SCK flow is continuous when MIDI[3:0] = 0).
2. CPHA = 0, CPOL = 0, SSIOP = 0, LSBFRST = 0.

Additionally, setting the SSOM bit invokes a specific mode, which interleaves pulses between data frames if there is a sufficient space to provide them (MIDI[3:0] must be set greater than one SPI period). Some configuration examples are shown in *Figure 974*.

Figure 974. NSS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0)



1. MSSI[3:0] = 0010, MIDI[3:0] = 0010.
2. NSS interleaves between data when MIDI[3:0] > 1 wide of the interleaving pulse is always one SCK period less than the gap provided between the frames (defined by the MIDI parameter). If MIDI is set, the frames are separated by a single SCK period but no interleaving pulse appears on NSS.

48.4.8 Ready pin (RDY) management

The status of the slave capability to handle data can be checked on the RDY pin. By default, a low level indicates that the slave is not ready for transfer. The reason can be that the slave TxFIFO is empty, RxFIFO full or the SPI is disabled. An active level of the signal can be selected by the RDIOP bit. If the master continues or starts to communicate with the slave when it indicates a not ready status, it is highly probable that the transfer fails.

The logic to control the RDY output is rather complex, tied closely with the TSIZE and DSIZE settings. The RDY reaction is more pessimistic and sensitive to TxFIFO becoming nearly empty and/or RxFIFO nearly full during a frame transfer. This pessimistic logic is suppressed at the end of a transfer only when RDY stays active, despite TxFIFO becomes fully empty and/or RxFIFO becomes fully occupied. The target is to prevent any data corruption and inform the master in time that it is necessary to suspend the transfer temporarily until the next transferred data can be processed safely again. When the RDY signal input is enabled at master side, the master suspends the communication once the slave indicates not ready status. This prevents the master to complete the transfer of an ongoing frame, which just empties the slave TxFIFO or full fills its RxFIFO until a next data is written and/or read there (despite the frame still can be completed without any constraint). It can make a problem if the TSIZE = 0 configuration is applied at slave because slave then never evaluates the end of the transfer (which suppresses the not ready status just when the last data is sent). Then the user has to release the RxFIFO and/or write additional (even dummy) data to TxFIFO by software at slave side to release the not RDY signal, unblock ST master and so enable it to continue at the communication suspended at middle of a frame occasionally.

When RDY is not used by the master, it must be disabled (RDIOM = 0). Then an internal logic of the master simulates the slave status always ready. In this case, the RDIOP bit setting has no meaning.

Due to synchronization between clock domains and evaluation of the RDY logic on both master and slave sides, the RDY pin feature is not reliable and cannot be used when the size of data frames is configured shorter than 8 bits.

48.4.9 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity, and the data frame format. To be able to communicate together, the master and slave devices must follow the same communication format and be synchronized correctly.

Clock phase and polarity controls

Four possible timing relationships can be chosen by software, using the CPOL and CPHA bits in the SPI_CFG2 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transferred (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transferred (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

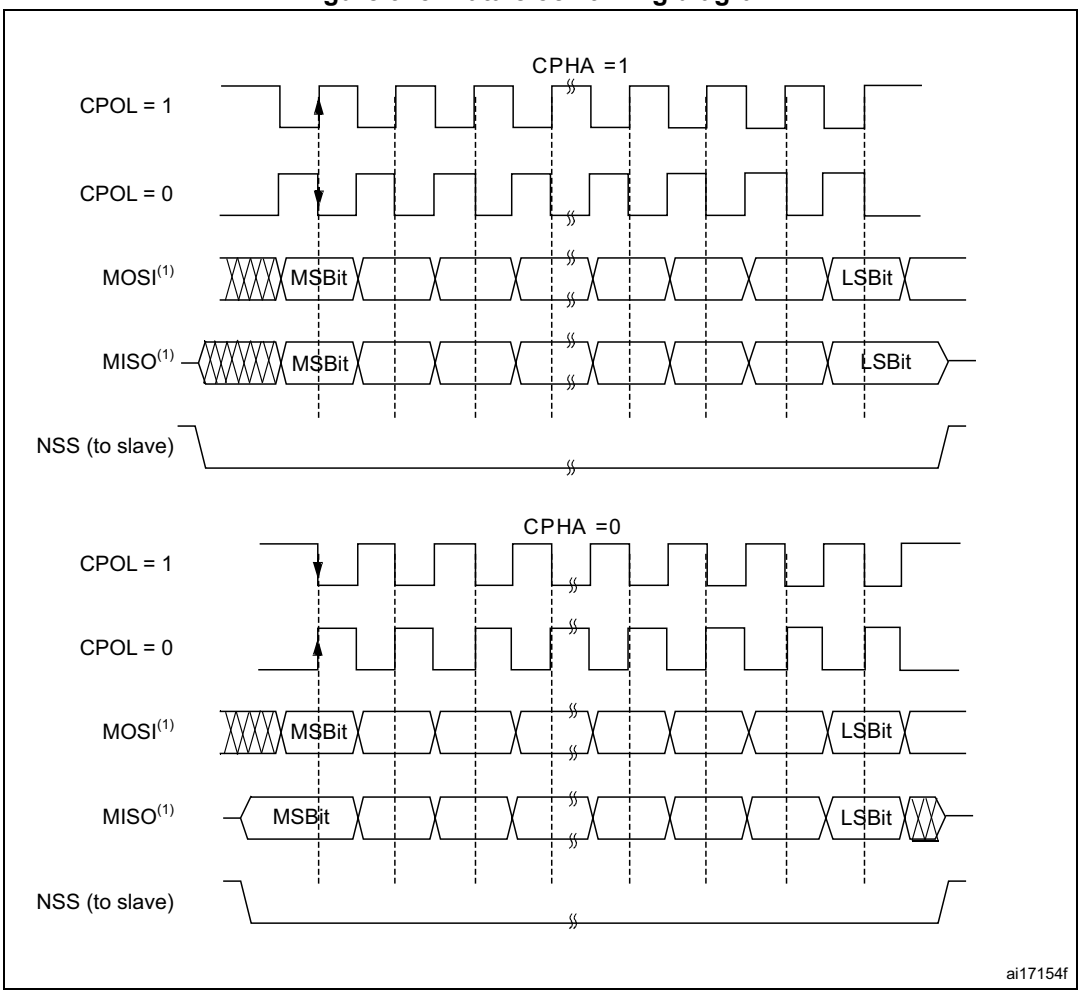
The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edges (dotted lines in *Figure 975*).

Figure 975 shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.

The idle state of SCK must correspond to the polarity selected in the SPI_CFG2 register (by pulling the SCK pin up if CPOL = 1 or pulling it down if CPOL = 0).

Figure 975. Data clock timing diagram



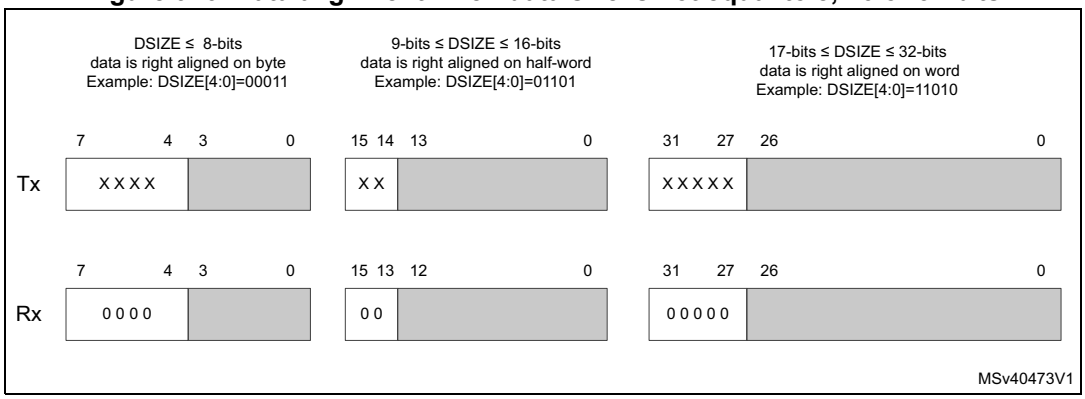
1. The order of data bits depends on the LSBFRST bit setting.
2. The input data sampling can be postponed by a half SCK cycle on the master MISO input (refer to the description of the DRDS bit of the SPI_CFG1 register).

Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFRST bit of the SPI_CFG2 register.

The data frame size is configured through the DSIZE bitfield of the SPI_CFG1 register to a range that depends on the SPI instance (see *Section 48.3: SPI implementation*). The setting applies to both transmission and reception. The data bits in the SPI_TXDR/SPI_RXDR registers are right-aligned with 8, 16, or 32 bits, depending on the data size (see *Figure 976*). These registers can consequently be accessed by bytes, half-words, or words. FIFO accesses smaller than a single data are forbidden. When the FIFO occupancy flag is raised, a FIFO access greater than the FIFO threshold can lead to spurious data being read or written data being lost. When the access to/from the SPI_TXDR/SPI_RXDR registers is a multiple of the configured data size, data packing is applied automatically, while the lowest significant bits/bytes are communicated first. For more details, see *Section 48.4.12: SPI data transmission and reception procedures*.

Figure 976. Data alignment when data size is not equal to 8, 16 or 32 bits



48.4.10 Configuring the SPI

The configuration procedure is almost the same for the master and the slave. For specific mode setups, follow the dedicated chapters. When a standard communication must be initialized, perform these steps:

1. Write the proper GPIO registers: configure GPIO alternate functions at MOSI, MISO, SCK, NSS, and RDY pins if applied.
2. Write into the SPI_CFG1 and SPI_CFG2 registers and set up the proper values of all ‘not reserved’ bits and bitfields, prior to enabling the SPI, with the following exceptions:
   a) The SSOM, MASRX, SSOE, RDIOM, MBR[2:0], BPASS, MIDI[3:0], MSSI[3:0] bits are taken into account in master mode only, the MSSI[3:0] bits take effect when the SSOE bit is set, the RDIOP bit takes no effect when the RDIOM bit is not set in master mode. When the slave is configured in TI mode, the MBR[2:0] setting is also considered.
   b) UDRCFG is taken into account in slave mode only.
   c) CRCSIZE[4:0] bitfield must be configured if CRCEN is set.
   d) CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI, and MIDI are not required in TI mode.

e) Once the AFCNTR bit is set in the SPI_CFG2 register, all the SPI outputs start to be propagated onto the associated GPIO pins regardless of the peripheral enable. So, any later configuration changes of the SPI_CFG1 and SPI_CFG2 registers can affect the level of signals on these pins.

3. Write to the SPI_CR2 register to select the length of the transfer, if it is not known TSIZE must be programmed to zero.
4. Write to the SPI_CRCPOLY and into the TCRCINI, RCRCINI, and CRC33_17 bits of the SPI_CR1 register to configure the CRC polynomial and CRC calculation if needed.
5. Configure DMA streams dedicated for the SPI Tx and Rx in DMA registers if the DMA streams are used (see *Section 48.4.14: Communication using DMA (direct memory addressing)*).
6. Configure SSI, HDDIR, and MASRX in the SPI_CR1 register if required.
7. Program the IOLOCK bit in the SPI_CR1 register if the configuration protection is required (for safety).

48.4.11 Enabling the SPI

It is recommended to configure and enable the SPI slave before the master sends the clock. But there is no impact if the configuration and enabling procedure is done while traffic is ongoing on the bus, assuming that the NSS signal is managed by hardware at slave or kept inactive by the slave software when the software management of the NSS signal is applied (see *Section 48.4.7*). To prevent any risk of any data underrun, all the data to be sent have to be written to the slave transmitter data register before the master starts its clocking. The SCK signal must be settled to the idle state level corresponding to the selected polarity, before the SPI slave is selected by NSS, else the following transfer may be desynchronized.

When the SPI slave is enabled at the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (its transition from nonactive to active level) just synchronizing the slave with the master. That is why the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection at the slave select software management. In this case, the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.

The master in full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled, the CSTART bit is set, and the TxFIFO is not empty, or with the next write to TxFIFO.

In any master receive-only mode, the master starts to communicate and the clock starts running after the SPI is enabled and the CSTART bit is set.

For handling DMA, see *Section 48.4.14*.

48.4.12 SPI data transmission and reception procedures

The SPI can transfer data at a very high communication speed. Even though the data is FIFO buffered, handling a continuous data flow by servicing data frames individually leads to an enormous CPU or DMA load, in particular when the data size is small. This potentially leads to a significant limitation of the overall system performance, as well as communication errors such as data overrun or underrun that may be raised when the system latencies in servicing requests become comparable to single data frame transfer time.

The SPI offers advanced hardware control features to prevent these issues from occurring:

- Decreasing the number of events requiring service by collecting the data frames into larger **data packets**

  The number of data frames per packet (FIFO threshold) can be configured. The complete data packet generates a FIFO occupancy event, which is then handled within a single, compact, service.
- Decreasing the number of CPU execution cycles required for the service, by applying the appropriate **data access**:
  - The widest access to the SPI data registers must be applied. This allows cumulative data to be handled by a single register read or write operation (data packing).
  - The number of read/write accesses necessary to complete the data packet must be aligned with the data frame size and the FIFO threshold.
- **Concurrent read and write services** to handle full-duplex data flow based on common FIFO occupancy events.
- Embedded **hardware data counters** to define the exact number of data involved in transfers.

Note: Using these features is optional. Nothing prevents the application from handling the data flow frame by frame.

The following sections give more details and describe specific cases for using these advanced hardware control features to handle data transfers.

Data packet control

The data frame size (number of bits in the frame) is defined through the DSIZE bitfield of the SPI_CFG1 register. The number of data frames per packet can be configured by selecting the FIFO threshold through the FTHLV bitfield of the SPI_CFG1 register. If the threshold value is set to zero, each completed data frame raises the FIFO occupancy event. The data packet occupancy must not exceed half of the FIFO size, which depends on the SPI instance of the product. The FIFO capacity (multiple of 8 bits) is consumed according to the following rules, depending on the data frame size:

- Data frames from 4 to 8 bits occupy one byte of the FIFO.
- Data frames from 9 to 16 bits occupy two bytes of the FIFO.
- Data frames from 17 to 24 bits occupy three bytes of the FIFO.
- Data frames from 25 to 32 bits occupy four bytes of the FIFO.

For example, if the FIFO capacity is 16 bytes, it accommodates up to five 24-bit frames. In this case, the data packet configuration must not exceed two data frames.

The SPI features two separate FIFOs to handle the reception and transmission data flow, the RxFIFO, and the TxFIFO. Their content can be handled by monitoring the occupancy

flags RXP, TXP, and DXP of the SPI_SR register according to the SPI mode (duplex or simplex):

- If RXP is set, at least one complete data packet can be read from the RxFIFO.
- If TXP is set, at least one complete data packet can be written to the TxFIFO.
- If DXP is set, at least one complete data packet can be read from the RxFIFO and written to the TxFIFO at full-duplex mode.

These flags can be polled by software, or they can trigger an interrupt and/or a DMA request (if enabled through the RXPIE, TXPIE, and DXPIE bits of the SPI_EIR register or the RXDMAEN and TXDMAEN bits of the SPI_CFG1 register).

Once an occupancy flag is set, the software or the DMA must read and/or write a complete data packet before checking the flag again to verify if the next packet can be handled. This cycle can be repeated until the corresponding flag is read at zero.
Both RxFIFO and TxFIFO contents are flushed and cannot be accessed when the SPI is disabled (SPE cleared in the SPI_CR1 register).

Data packing versus data register access control

The content of the RxFIFO and TxFIFO can be accessed by reading/writing from/to the SPI data registers SPI_RXDR and SPI_TXDR, respectively. A read access from the SPI_RXDR register returns the oldest values stored in the RxFIFO that have not been read yet. A write access to the SPI_TXDR register stores the data written at the end of the send queue in the TxFIFO.

These data registers can be accessed by 8-, 16-, or 32-bit read and write CPU instructions, forced by the register address casting applied by the software. Data packing is performed automatically by hardware if the data access applied by the software is a multiple of the data size. It allows handling more than one data in parallel in a single data register access. Then the SPI operates using the lowest significant byte or half-word first. FIFO data accesses of less than the configured data size are forbidden. To avoid spurious data being read or written data being lost, a complete data packet (configured through the FIFO threshold) must be serviced when the corresponding FIFO occupancy flag is set. If the data pattern is not byte-, half-word, or word-aligned, only the valid data bits are stored right-aligned to the FIFO and transferred on the bus. Unused bits are discarded on the transmitter side and padded with zeros on the receiver side.

For example, if the data frame size fits into one byte, the data packing is used automatically when a 16-bit or 32-bit read/write access is performed by software from/to the SPI_RXDR/SPI_TXDR register. In this case, two respectively four data are handled by a single 16-bit respectively 32-bit access. Additionally, if such data frames are grouped into packets of four via the FIFO threshold setting, the packet to be serviced is signalized by raising the corresponding FIFO occupancy flag (TXP, RXP, or DXP). If the instance FIFO threshold is sufficient, a packet containing eight 8-bit data frames can be handled by two consecutive 32-bit accesses upon the same single threshold event. The most efficient data handling is achieved when the data packet size (defined by DSIZE and FTHLV bitfields) is aligned with the read/write access from/to the data registers.

Concurrent read and write services

In full-duplex mode, both TxFIFO and RxFIFO packet occupancies can be monitored through a common FIFO flag (DXP). When the DXP flag is set, the application performs the specified number of writes to SPI_TXDR to upload the content of one entire data packet for transmission, followed by the same number of reads from SPI_RXDR to download the

content of one received data packet. Once one data packet is uploaded and one packet is downloaded, the application software or the DMA checks again the DXP flag and repeats the data packet read and write operation sequence until DXP is read as zero.

The drawback of services based on DXP exclusively is that servicing the TxFIFO is delayed on purpose due to the SPI nature since the TXP events precedes the RXP ones. To allow continuous SCK clock flow on the master side and prevent underrun on the slave side, it is recommended to prefill a few data ahead to the TxFIFO at the transfer start, and wait until the transfer is complete to read the last received data.

Hardware data counter

If a hardware data counter is used (TSIZE bitfield of the SPI_CR2 register set to a nonzero value), the application software does not need to calculate the remaining number of data to be handled. The end of transfer automatically controls the CRC, as well as the hardware slave select management, when used. The user application does not need to ensure that the overall number of data is a multiple of the packet size. If the last data packet is incomplete, it is serviced in the same way as any full packet. The unused part of this last packet is not handled by the peripheral. Only the valid data are written into the TxFIFO and/or read from the RxFIFO, and the redundant writes and reads are discarded. When the hardware counter reaches zero, the EOT flag is raised, and the RXP flag is not set for the last data packet. If the last packet is not aligned with the packet size, the TXP and EOT occupancy events are not related to the configured packet size but to the number of remaining data calculated by hardware.

If TSIZE is kept at zero (for example due to an unpredictable number of data), only the number of transferred data corresponding to the FIFO thresholds is supported. If some data are not aligned with the configured packet size, they may remain pending and available in the RxFIFO. In this case, the FTHLV level is not reached and the RXP flag is not set. Then the number of remaining received data frames in the RxFIFO is indicated by the RXWNE and RXPLVL bitfields of the SPI_SR register. Nevertheless, the application software can still read the complete data packet from the RxFIFO and the redundant data are read as zero. To prevent such an unaligned data reception, the user must configure the FIFO threshold to a single data (FTHLV = 0). In this case, each data frame is serviced by its own RXP occupancy event.

Data transfer handling

Data are transferred using MOSI and MISO lines, depending on the SPI communication mode and associated configuration. The mode is configured through the COMM[1:0] bitfield of the SPI_CFG2 register. The HDDIR bit of the SPI_CR1 register controls the data flow direction in half-duplex mode. The communication flow is handled by the master via the NSS and SCK signals. The slave selected for the communication is fully subordinated to the master communication activity, no matter if it handles the data flow on time or not. The slave can only temporarily suspend the master communication by using the RDY signal. The active levels of RDY and NSS signals can be changed, or MISO and MOSI functionality swapped (via the RDIOP, SSIOP and IOSWP bits of the SPI_CFG2 register).

The SPI master transmitter can operate in full-duplex, simplex, or half-duplex mode. In full-duplex mode, data are received synchronously. The master starts the data transfer once the CSTART bit of the SPI_CR1 register is set, provided the SPI is enabled and the TxFIFO

content is not empty. The master then provides the serial clock signal continuously on the SCK pin until:

- The total number of required data programmed in TSIZE is transferred, or
- The transfer is suspended.

An automatic temporary suspension of the master transmission occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The TxFIFO becomes empty, or
- In full-duplex mode, the RxFIFO becomes full while the automatic suspension is enabled (MASRX set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock, and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA writes additional data to TxFIFO.
- In full-duplex mode, the master software or the DMA releases the RxFIFO to enable it to accommodate new data.

The SPI master can receive data only when it operates in simplex receiver or half-duplex receiver mode. In these modes, the master starts the data transfer when the SPI is enabled, and the transfer is released by setting the CSTART bit. The serial clock signal is then provided continuously on the SCK pin by the master until:

- The total number of required data programmed in TSIZE is received, or
- The transfer is suspended by the master.

An automatic temporary suspension of the reception occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The RxFIFO becomes full while the automatic suspension is enabled (MASRX bit set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA releases the RxFIFO to enable it in order to accommodate new data.

A preferable way to terminate a transfer is to program the TSIZE bitfield to generate an EOT event. Hardware NSS signal or CRC handshake can then be controlled. To restart the internal state machine properly, it is recommended to disable the SPI and enable it again when the transfer is complete, even if the SPI configuration is not changed.

A transfer can be suspended at any time by setting the CSUSP bit of the SPI_CR1 register, which clears the CSTART bit. This software suspension control ensures the completion of any ongoing data frame. To restart the internal state machine properly, the SPI must be disabled and enabled again before the next transfer starts.

Oppositely, a temporary automatic suspension controlled by hardware typically results in a frozen and incomplete data frame transfer. This depends on baud rate setting, but usually, due to internal synchronization delays, the SCK signal stops when a few bits from the next data frame are already transferred on the bus. Once the suspension is released, the data frame transmission completes by transferring the remaining bits. That is why this automatic suspension is not quite reliable when the data frame size is less than eight bits. When shorter data frames are used, to prevent any data loss and assure proper RDY and/or

MASRX operation, the user can interleave and so extend the transfer time by inserting additional dummy clock cycles so that the period for a single data frame is higher or equal to eight SCK duration. This is done by setting the MIDI[3:0] bitfield of the SPI_CFG2 register.

Caution: If the SPE bit is cleared in master mode while the transfer is ongoing without any suspension, the clock is stopped even if the current frame transmission is not complete, and the content of the FIFOs is flushed and lost.

48.4.13 Disabling the SPI

To disable the SPI, it is mandatory to follow the disable procedures described in this paragraph.

In the master mode, it is important to do this before the system enters a low-power mode when the peripheral clock is stopped, otherwise, ongoing transfers may be corrupted.

In slave mode, the SPI communication can continue when the **spi_pclk** and **spi_ker_ck** clocks are stopped, without interruption, until any end of communication or data service request condition is reached. The **spi_pclk** can generally be stopped by setting the system into Stop mode. Refer to the RCC section for further information.

The master in full-duplex or transmit-only mode can finish any transfers when it stops providing data for transmission. In this case, the clock stops after the last data transfer. TXC flag can be polled (or interrupt enabled with EOTIE = 1) in order to wait for the last data frame to be sent.

When the master is in any receive-only mode, to stop the peripheral, the SPI communication must first be suspended, by setting the CSUSP bit.

The data received but not read remain stored in RxFIFO when the SPI is suspended.

After such a software suspension, SPI must always be disabled to restart the internal state machine properly.

When SPI is disabled, RxFIFO is flushed. To prevent losing unread data, the user must ensure that RxFIFO is empty when disabling the SPI, by reading all remaining data (as indicated by the RXP, RXWNE, and RXPLVL fields in the SPI_SR register).

The standard disable procedure is based on polling EOT and/or TXC status to check if a transmission session is (fully) completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transfers, for example:

- When the master handles the NSS signal by a GPIO not related to SPI (for example at case of multislave star topology) and it has to provide a proper end-of-NSS pulse for the slave, or
- When transfers from DMA or FIFO are completed while the last data frame or CRC frame transfer is still ongoing in the peripheral bus.

When TSIZE > 0, EOT and TXC signals are equal so polling of EOT is reliable at whatever SPI communication mode to check the end of the bus activity. When TSIZE = 0, the user has to check TXC, SUSP, or FIFO occupancy flags according to the applied SPI mode and the way of the data flow termination.

The correct disable procedure in master mode, except when receive-only mode is used, is:

1. Wait until TXC = 1 and/or EOT = 1 (no more data to transmit and last data frame sent). When CRC is used, it is sent automatically after the last data in the block is processed.

TXC/EOT is set when the CRC frame is complete in this case. When a transmission is suspended the software has to wait until the CSTART bit is cleared.

2. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
3. Disable the SPI (SPE = 0).

The correct disable procedure for master receive-only modes is:

1. Wait on EOT or break the receive flow by suspending SPI (CSUSP = 1).
2. Wait until SUSP = 1 (the last data frame is processed) if the receive flow is suspended.
3. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
4. Disable the SPI (SPE = 0).

In slave mode, any ongoing data are lost when disabling the SPI.

Controlling the I/Os

As soon as the SPI is disabled, the associated and enabled AF outputs can still be driven by the device depending on the AFCNTR bit of the SPI_CFG2 register. When active output control is applied (AFCNTR = 1) and SPI has just been disabled (SPE = 0), the enabled outputs associated with SPI control signals (like NSS and SCK at master and RDY at slave) can immediately toggle to inactive level (according to SSIOP and CPOL settings at master and RDIOP at slave respectively). Instead, the data line output (MOSI at master and MISO at slave) can immediately change its level depending on the actual TxFIFO content, with the effect of potentially making invalid and no more guaranteed the value of the latest transferred bit on the bus. If necessary, the user has to take care about proper data hold time at the data line and avoid any eventual fast SPI disable just after the last data transfer is complete.

Note: Despite stability of the latest bit is guaranteed by design during the sampling edge of the clock, some devices can require even extension of this data bit stability interval during the sampling. It can be done, for example by inserting a small software delay between EOT event occurrence and SPI disable action.

48.4.14 Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXDMAEN or RXDMAEN enable bits of the SPI_CFG1 register are set. Separate requests must be issued to the Tx and Rx buffers to fulfill the service of the defined packet.

- In transmission, a series of DMA requests is triggered each time TXP is set. The DMA then performs a series of writes to the SPI_TXDR register.
- In reception, a series of DMA requests is triggered each time RXP is set. The DMA then performs a series of reads from the SPI_RXDR register. When EOT is set at the end of the transfer and the last data packet is incomplete, then DMA request is activated automatically according to RXWNE and RXPLVL[1:0] setting to read the rest of data.

If the SPI is programmed in receive-only mode, UDR is never set.

If the SPI is programmed in a transmit mode, TXP and UDR can be eventually set at slave side, because transmit data may not be available. In this case, some data are sent on the TX line according with the UDR management selection.

When the SPI is used at a simplex mode, the user must enable the adequate DMA channel only while keeping the complementary unused channel disabled.

If the SPI is programmed in transmit-only mode, RXP and OVR are never set.

If the SPI is programmed in full-duplex mode, RXP and OVR are eventually set, because received data are not read.

In transmission mode, when the DMA or the user has written all the data to be transmitted (the TXTF flag is set in the SPI_SR register), the EOT (or TXC at case TSIZE = 0) flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or before disabling the **spi_pclk** in master mode. The software must first wait until EOT = 1 and/or TXC = 1.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable the DMA Rx buffer in the RXDMAEN bit of the SPI_CFG1 register, if DMA Rx is used.
2. Enable DMA requests for Tx and Rx in DMA registers, if the DMA is used.
3. Enable the DMA Tx buffer in the TXDMAEN bit of the SPI_CFG1 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication, it is mandatory to follow these steps in order:

1. Disable DMA requests for Tx and Rx in the DMA registers, if the DMA issued.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits of the SPI_CFG1 register, if DMA Tx and/or DMA Rx are used.

Data packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPI_CFG1 register) the packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel.

The packing mode is enabled if the DMA channel PSIZE value is a multiple of the data size. Then the DMA automatically manages the sequences of write and read operations to/from the SPI data registers, based on FIFO occupancy flags, and depending on the FIFO threshold and data size configurations.

The DMA completes the transfer automatically according to the TSIZE field setting, whatever the data packing mode used, and even if the number of data to transfer is not a multiple of the DMA data size (16 bits or 32 bits) while the frame size is smaller.

Alternatively, the last data frames can be written by software, in the single/unpacked mode.

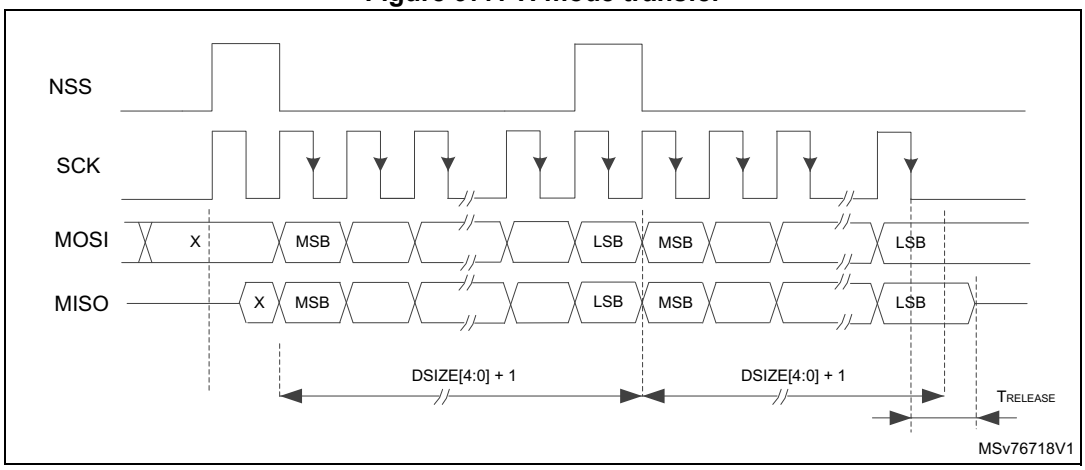
Configuring any DMA data access to less than the configured data size is forbidden. One complete data frame must be always accessed at minimum.

48.5 SPI specific modes and control

48.5.1 TI mode

With a specific SP[2:0] bitfield setting of the SPI_CFG2 register, the SPI can be configured compliant with the TI protocol. The SCK and NSS signals polarity, phase and flow, as well as the bit order are fixed, so the setting of CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI, and MIDI is not required when the SPI is in TI mode configuration. The NSS signal synchronizes the protocol by pulses over the LSB data bit as it is shown in *Figure 977*.

Figure 977. TI mode transfer



In slave mode, the clock generator is used to define the time when the slave output at MISO pin becomes high-Z when the current transfer finishes. The master baud rate setting (MBR[2:0] of SPI_CFG1) is applied and any baud rate can be used to determine this moment with optimal flexibility. The delay for the MISO signal to become high-Z (TRELEASE) depends on internal resynchronization, too, which takes the next additional 2-4 periods of the clock signal feeding the generator. It is given by the following formula:

$$\frac{Tbaud}{2} + 2 \times Tspi_ker_ck \leq Trelease \leq \frac{Tbaud}{2} + 4 \times Tspi_ker_ck$$

If the slave detects a misplaced NSS pulse during a data transfer the TIFRE flag is set.

48.5.2 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and the interrupt is enabled by setting the corresponding interrupt enable bit.

Overrun flag (OVR)

An overrun condition occurs when data are received by a master or slave and the RxFIFO has not enough space to store these received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RxFIFO).

When an overrun condition occurs, the OVR flag is set and the newly received value does not overwrite the previous one in the RxFIFO. The newly received value is discarded and all

data transmitted subsequently are lost. The OVR flag triggers an interrupt if the OVRIE bit is set. Clearing the OVR bit is done by setting the OVRC bit of the SPI_IFCR register. Clearing the RxFIFO content by performing software reads before the OVR bit is cleared reduces the risk of any immediate repetition of its next overrun. It is suggested to release the RxFIFO space as much as possible, this means to read out all the available data packets based on RXP flag indication.

In master mode, the user can prevent the RxFIFO overrun by automatic communication suspend (MASRX bit).

Underrun flag (UDR)

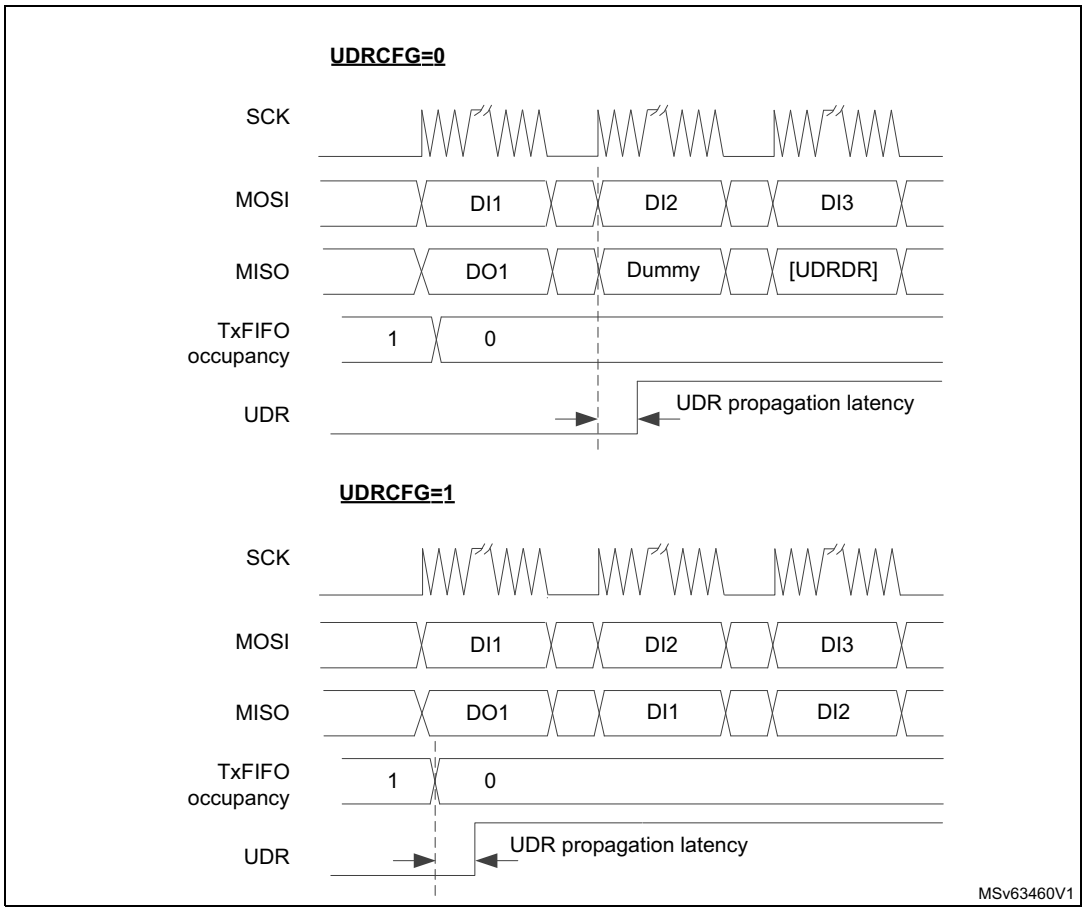
In a slave-transmitting mode, the underrun condition is captured internally by hardware if no data is available for transmission in the slave TxFIFO commonly. The UDR flag setting is then propagated into the status register by hardware (see note below). UDR triggers an interrupt if the UDRIE bit is set.

Underrun detection logic and system behavior depend on the UDRCFG bit. When an underrun is detected by the slave, it can provide out either a constant pattern stored by the user at the UDRDR register or the data received previously from the master. When the first configuration (UDRCFG = 0) is applied, the underrun condition is evaluated whenever the master starts to communicate a new data frame while TxFIFO is empty. Then single additional dummy (accidental) data is always inserted between the last valid data and the constant pattern defined at the UDRDR register (see *Figure 978*). The second configuration (UDRCFG=1) can be used in circular topography structures (see *Figure 970*). Assuming that TxFIFO is not empty when the master starts the communication, the underrun condition is evaluated just once the FIFO becomes empty during the next data flow. Valid data from TxFIFO is then upended by the last received data immediately.

The standard transmission is reenabled once the software clears the UDR flag and this clearing is propagated into SPI logic by hardware. Writing some data to the TxFIFO before the UVR bit is cleared reduces the risk of any immediate repetition of its next underrun.

The data transferred by the slave is unpredictable especially when the transfer starts or continues while TxFIFO is empty and the underrun condition is either not yet captured or just cleared. Typically, this is the case when SPI is just enabled or when a transfer with a defined size just starts. First bits can be corrupted in this case, as well, when the slave software writes the first data into the empty TxFIFO too close prior to starting the data transfer (propagation of the data into TxFIFO takes a few APB clock cycles).

Figure 978. Optional configurations of the slave behavior when an underrun condition is detected



Note: The hardware propagation of an UDR event needs additional traffic on the bus. It always takes a few extra SPI clock cycles after the event happens (both underrun captured by hardware and cleared by software). If clearing of the UDR flag by software is applied close to the end of the data frame transfer or when the SCK line is at idle in between the frames, the next extra underrun pattern is sent initially by the slave before the valid data from TxFIFO becomes transferred again. The user can prevent this by SPI disable/enable action between sessions to restart the underrun logic and so initiate the next session by the valid data.

Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in slave select hardware mode, or SSI bit in slave select software mode) pulled low. This automatically affects the SPI interface in the following ways:

- The MODF bit is set and the SPI interrupt is triggered if the MODFIE bit is set.
- The SPE bit is forced to zero until the MODF bit is set. This disables the SPI and blocks all the peripheral outputs except the MODF interrupt request if enabled.
- The MASTER bit is cleared, thus forcing the device into slave mode.

MODF is cleared by writing 1 to the MODFC bit of the SPI_IFCR register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled to its nonactive level before reenabling the SPI, by setting the SPE bit.

As a security, the hardware does not allow the SPE bit to be set while the MODF bit is set. In a slave device, the MODF bit cannot be set except as the result of a previous multimaster conflict.

A correct software procedure when a master overtakes the bus in multimaster systems must be the following one:

1. Switch into master mode while SSOE = 0 (potential conflict can appear when another master occupies the bus. In this case, MODF is raised, which prevents any next node switching into master mode).
2. Put GPIO pin dedicated for another master slave select control into active level.
3. Perform a data transfer.
4. Put GPIO pin dedicated for another master slave select control into nonactive level.
5. Switch back to slave mode.

CRC error (CRCE)

This flag is used to verify the validity of the value received when the CRCEN bit of the SPI_CFG1 register is set. The CRCE flag of the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRC value, after the last data is received (as defined by TSIZE). The CRCE flag triggers an interrupt if the CRCEIE bit is set. Clearing the bit CRCE is done by a writing 1 to the CRCEC bit of the SPI_IFCR register.

TI mode frame format error (TIFRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the TIFRE flag is set in the SPI_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of a few data frames.

The TIFRE flag is cleared by writing 1 to the TIFREC bit of the SPI_IFCR register. If the TIFREIE bit is set, an interrupt is generated on the NSS error detection. As data consistency is no longer guaranteed, communication must be reinitiated by software between master and slave.

48.5.3 CRC computation

Two separate CRC calculators are implemented to check the reliability of transmitted and received data. The CRC polynomial configuration depends on the instance. Refer to *Section 48.3: SPI implementation* for more information.

The length of the polynomial is defined by the most significant bit of the value stored in the SPI_CRCPOLY register. It must be greater than the data frame size (in bits) defined in the DSIZE[4:0] bitfield of the SPI_CFG1 register. To obtain a full-size polynomial, the polynomial length must exceed the maximum data size of the peripheral instance, and the CRC33_17 bit of the SPI_CR1 register must be set to select the most significant bit of the polynomial string.

For example, to select the standard CRC16-CCITT (XMODEM) polynomial x^16 + x^12 + x^5 + 1:

- For a 32-bit instance: write 0x11021 to the SPI_CRCPOLY register and clear the CRC33_17 bit.
- For a 16-bit instance: to obtain the full size, write 0x1021, and set the CRC33_17 bit.

The CRCSIZE field of the SPI_CFG1 register then defines how many most significant bits from the CRC calculation registers are transferred and compared as CRC frame. It is defined independently from the data frame length, but it must be either equal or an integer multiple of the data frame size. Its size cannot exceed the maximum data size of the instance.

To fully benefit from the CRC calculation capability, the polynomial length setting must correspond to the CRC frame size, else the bits unused at the calculation are transferred and expected all zero at the end of the CRC frame if its size is set greater than the polynomial length.

CRC principle

The CRC calculation is enabled by setting the CRCEN bit of the SPI_CFG1 register before the SPI is enabled (SPE = 1). The CRC value is then calculated using the CRC polynomial defined by the CRCPOLY register and CRC33_17 bit. When SPI is enabled, the CRC polynomial can be changed but only in the case when there is no traffic on the bus.

The CRC computation is done, bit by bit, on the sampling clock edge defined by the CPHA and CPOL bits of the SPI_CR1 register. The calculated CRC value is checked automatically at the end of the data block defined by the SPI_CR2 register exclusively.

When a mismatch is detected between the CRC calculated internally on the received data and the CRC received from the transmitter, a CRCE flag is set to indicate a data corruption error. The right procedure for handling the CRC depends on the SPI configuration and the chosen transfer management.

CRC transfer management

Communication starts and continues normally until the last data frame has been sent or received in the SPI_DR register.

The length of the transfer must be defined by TSIZE. When the desired number of data is transferred, the TXCRC is transmitted and the data received on the line are compared to the RXCRC value.

Whatever the CRCSIZE configuration, TSIZE cannot be set to 0xFFFF for a full-featured instance, and to 0x3FF for a limited-featured instance, if CRC is enabled.

In transmission, the CRC computation is frozen during the CRC transfer and the TXCRC is transmitted, in a frame of length equal to the CRCSIZE field value.

In reception, the RXCRC is also frozen when the desired number of data is transferred. Information to be compared with the RXCRC register content is then received in a frame of length equal to the CRCSIZE value.

Once the CRC frame is complete, an automatic check is performed comparing the received CRC value and the value calculated in the SPI_RXCRC register. The software has to check the CRCE flag of the SPI_SR register to determine if the data transfers were corrupted or not. Software clears the CRCE flag by writing 1 to the CRCEC.

The user takes no care about any flushing redundant CRC information, it is done automatically.

Resetting the SPI_TXCRC and SPI_RXCRC values

The SPI_TXCRC and SPI_RXCRC values are initialized automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode to transfer data without any interruption (several data blocks covered by intermediate CRC checking phases). Initialization patterns for receiver and transmitter can be configured either to zero or to all ones in dependency on setting bits TCRCINI and RCRCINI of the SPI_CR1 register.

There is no need to reset these registers between communications, and they are not cleared when SPI is disabled. Their contents are not reliable while communication is ongoing; they become reliable only after the CRCE or EOT event is asserted, in particular.

48.6 SPI in low-power modes

Table 556. Effect of low-power modes on the SPI

Mode	Description
Sleep	No effect. SPI interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the SPI registers is kept.
Standby	The SPI instance is not functional in this mode. It is powered down, and must be reinitialized after exiting Standby mode.

1. Refer to *Section 48.3: SPI implementation* for information about wake-up from Stop mode support per instance as well as Standby mode availability. If an instance is not functional in a Stop mode, it must be disabled before entering this Stop mode.

48.7 SPI interrupts

Table 557 gives an overview of the SPI events capable of generating interrupts if enabled. Some of them feature wake-up from low-power mode capability, additionally. Most of them can be enabled and disabled independently while using specific interrupt enable control bits. The flags associated with the events are cleared by specific methods. Refer to the description of SPI registers for more details about the event flags. When SPI is disabled, all the pending interrupt requests are blocked to prevent their propagation into the interrupt services, except the MODF interrupt request.

Table 557. SPI wake-up and interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Event clear method	Exit from Stop and Standby modes capability[1][2]
SPI	TxFIFO ready to be loaded (space available for one data packet - FIFO threshold)	TXP	TXPIE	TXP cleared by hardware when TxFIFO contains less than FTHLV empty locations	Yes
	Data received in RxFIFO (one data packet available - FIFO threshold)	RXP	RXPIE	RXP cleared by hardware when RxFIFO contains less than FTHLV samples	Yes
	Both TXP and RXP active	DXP	DXPIE	When TXP or RXP are cleared	Yes
	Transmission Transfer Filled	TXTF	TXTFIE	Writing TXTFC to 1	No
	Underrun	UDR	UDRIE	Writing UDRC to 1	Yes
	Overrun	OVR	OVRIE	Writing OVRC to 1	Yes
	CRC Error	CRCE	CRCEIE	Writing CRCEC to 1	Yes
	TI Frame Format Error	TIFRE	TIFREIE	Writing TIFREC to 1	No
	Mode Fault	MODF	MODFIE	Writing MODFC to 1	No
	End Of Transfer (full transfer sequence completed - based on TSIZE value)	EOT	EOTIE	Writing EOTC to 1	Yes
	Master mode suspended	SUSP		Writing SUSPC to 1	Yes
	TxFIFO transmission complete (TxFIFO empty)	TXC[3]		TXC cleared by hardware when a transmission activity starts on the bus	No

1. All the interrupt events can wake up the system from Sleep mode at each instance. For detailed information about instances capabilities to exit from concrete Stop and Standby mode refer to the *Functionalities depending on the working mode* table.
2. Refer to *Section 48.3: SPI implementation* for information about Standby mode availability.
3. The TXC flag behavior depends on the TSIZE setting. When TSIZE>0, the flag fully follows the EOT one including its clearing by EOTC.

48.8 I2S main features

- Full duplex communication
- Simplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler
- Data length can be 16, 24 or 32 bits[a]
- Channel length can be 16 or 32 in master, any value in slave
- Programmable clock polarity
- Error flags signaling for improved reliability: Underrun, overrun, and frame errors
- Embedded Rx and TxFIFOs
- Supported I^2S protocols:
  - I^2S Philips standard
  - MSB-justified standard (left-justified)
  - LSB-justified standard (right-justified)
  - PCM standard (with short and long frame synchronization)
- Data ordering programmable (LSb or MSb first)
- DMA capability for transmission and reception
- Master clock can be output to drive an external audio component:
  - F_{MCK} = 256 x F_{WS} for all I2S modes
  - F_{MCK} = 128 x F_{WS} for all PCM modes

Note: F_{MCK} is the master clock frequency and F_{WS} is the audio sampling frequency.

48.9 I2S functional description

48.9.1 I2S general description

The block diagram shown on *Figure 965* also applies for I2S mode.

The SPI/I2S block can work in I2S/PCM mode, when the bit I2SMOD is set. A dedicated register (SPI_I2SCFGR) is available for configuring the dedicated I2S parameters, which include the clock generator, and the serial link interface.

The I2S/PCM function uses the clock generator to produce the communication clock when the SPI/I2S is set in master mode. This clock generator is also the source of the master clock output (MCK).

Resources such as RxFIFO, TxFIFO, DMA, and parts of interrupt signaling are shared with the SPI function. The low-power mode function is also available in I2S mode (refer to *Section 48.6: SPI in low-power modes* and *Section 48.10: I2S interrupts*).

a. Not always available, refer to *Section 48.3: SPI implementation* in order to check if 24 and 32-bit data widths are supported.

48.9.2 Pin sharing with SPI function

The I2S shares four common pins with the SPI:

- SDO: serial data output (mapped on the MOSI pin) to transmit the audio samples in master, and receive the audio sample in slave mode. Refer to *Section : Serial data line swapping on page 2913*.
- SDI: serial data input (mapped on the MISO pin) to receive the audio samples in master, and transmit the audio sample in slave mode. Refer to *Section : Serial data line swapping on page 2913*.
- WS: word select (mapped on the NSS pin). This is the frame synchronization. It is configured as output in master mode, and as input in slave mode.
- CK: serial clock (mapped on the SCK pin). This is the serial bit clock. It is configured as output in master mode, and as input in slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- MCK: master clock (mapped separately) is used when the I2S is configured in master mode.

48.9.3 Bitfields usable in I2S/PCM mode

When the I2S/PCM mode is selected (I2SMOD = '1'), some bitfields are no longer relevant, and must be forced to a specific value to guarantee the behavior of the I2S/PCM function. *Table 558* shows the list of bits and fields available in the I2S/PCM mode, and indicates which must be forced to a specific value.

Table 558. Bitfields usable in PCM/I2S mode

Register name	Bitfields usable in PCM/I2S Mode	Constraints on other bitfields
SPI/I2S control register 1 (SPI_CR1)	IOLOCK, CSUSP, CSTART, SPE	Other fields set to their reset values
SPI/I2S control register 2 (SPI_CR2)	-	Set to reset value
SPI/I2S configuration register 1 (SPI_CFG1)	TXDMAEN, RXDMAEN, FTHLV	Other fields set to their reset values
SPI/I2S configuration register 2 (SPI_CFG2)	AFCNTR, LSBFRST, IOSWP	Other fields set to their reset values
SPI/I2S interrupt enable register (SPI_IER)	TIFREIE, OVRIE, UDRIE, TXPIE, RXPIE	
SPI/I2S status register (SPI_SR)	SUSP, TIFRE, OVR, UDR, TXP, RXP	Other flags not relevant
SPI/I2S interrupt/status flags clear register (SPI_IFCR)	SUSPC, TIFREC, OVRC, UDRC	Other fields set to their reset values
SPI/I2S receive data register (SPI_RXDR)	The complete register	-

Table 558. Bitfields usable in PCM/I2S mode (continued)

Register name	Bitfields usable in PCM/I2S Mode	Constraints on other bitfields
SPI/I2S polynomial register (SPI_CRCPOLY)	-	Set to reset value
SPI/I2S transmitter CRC register (SPI_TXCRC)	-	
SPI/I2S receiver CRC register (SPI_RXCRC)	-	
SPI/I2S underrun data register (SPI_UDRDR)	-	
SPI/I2S configuration register (SPI_I2SCFGR)	The complete register	-

48.9.4 Slave and master modes

The SPI/I2S block supports master and slave modes for both I2S and PCM protocols. In master mode, both CK, WS and MCK signals are set to output.

In slave mode, both CK and WS signals are set to input. The signal MCK cannot be used in slave mode.

To improve the robustness of the SPI/I2S block in slave mode, the peripheral resynchronizes each reception and transmission on WS signal. This means that:

- In I2S Philips standard, the shift-in or shift-out of each data is triggered one bit clock after each transition of WS.
- In I2S MSB-justified standard, the shift-in or shift-out of each data is triggered as soon as a transition of WS is detected.
- In PCM short standard, the shift-in or shift-out of each data is triggered one bit clock after the active edge of WS.
- In PCM long standard, the shift-in or shift-out of each data is triggered as soon as the active edge of WS is detected

Note: This resynchronization mechanism is not available for the I2S LSB-justified standard.

Note as well that there is no need to provide a kernel clock when the SPI/I2S is configured in slave mode.

48.9.5 Supported audio protocols

The I2S/PCM interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits of the SPI_I2SCFGR register.

In the I2S protocol, the audio data are time-multiplexed on two channels: the left channel and the right channel. The WS signal is used to indicate which channel must be considered as the left, and which one is the right.

In I2S master mode, four frame formats are supported:

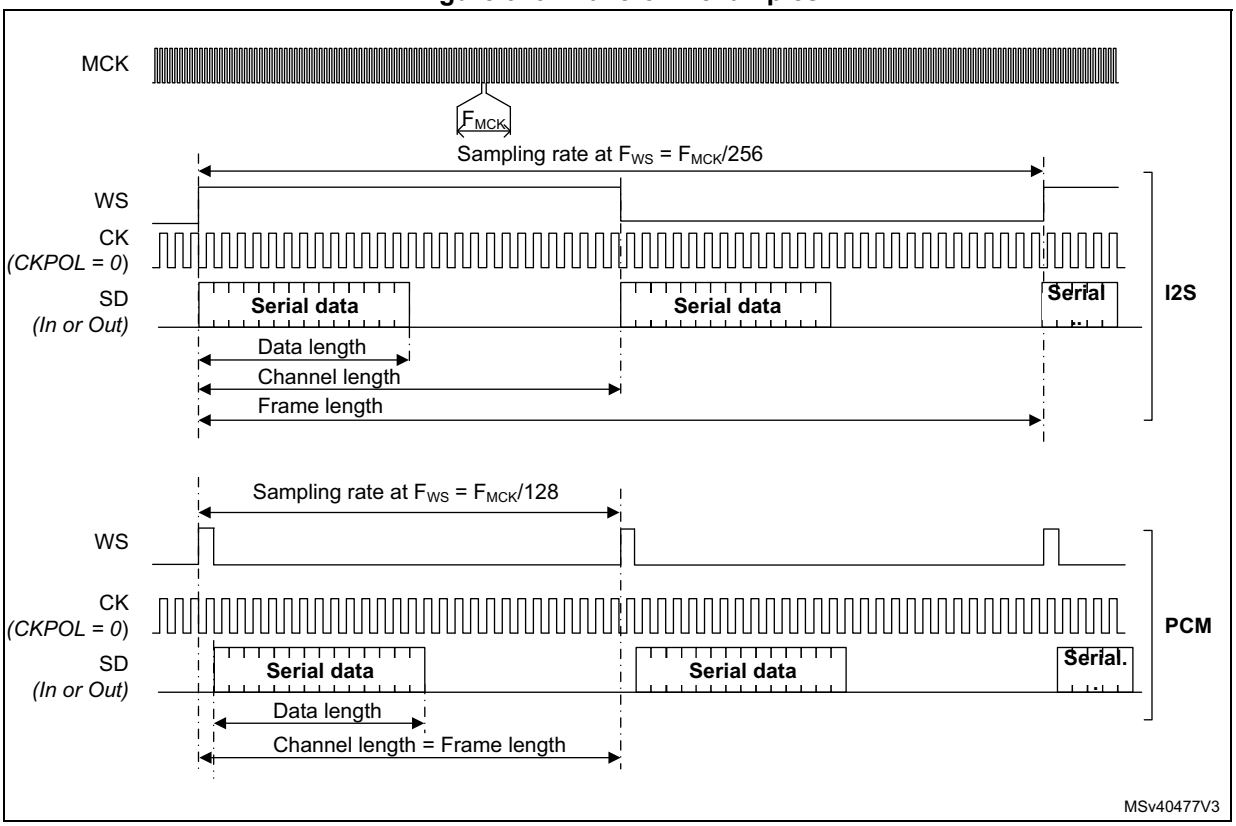
- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel[(a)]
- 32-bit data packed in a 32-bit channel[(a)]

In PCM master mode, three frame formats are supported:

- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel[(a)]

The figure hereafter shows the main definition used in this section: data length, channel length and frame length.

Figure 979. Waveform examples



For simplicity sake, in the next figures, SDI represents the serial data input and SDO the serial data output. Refer to *Section : Serial data line swapping* for details about the direction control of serial data lines.

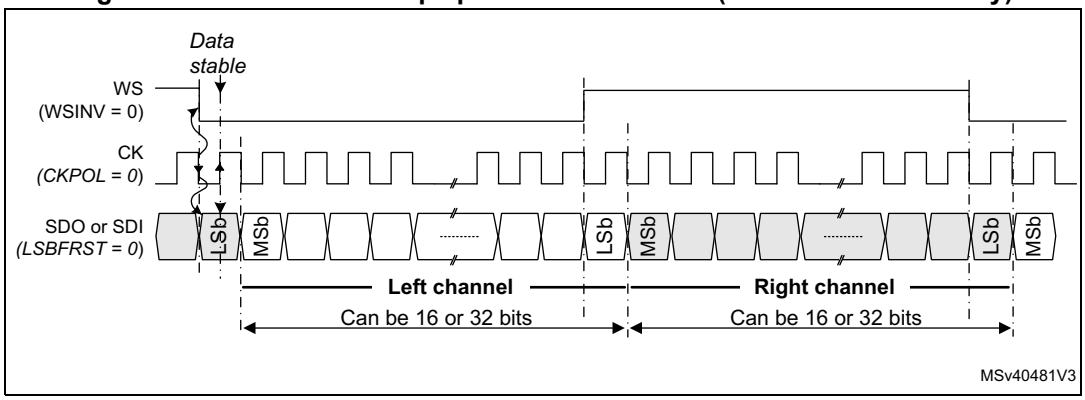
I²S Philips standard

The I2S Philips standard is selected by setting I2SSTD to 0b00. This standard is supported in master and slave mode.

In this standard, the WS signal toggles one CK clock cycle before the first bit (MSb in I2S Philips standard) is available. A falling edge transition of WS indicates that the next data transferred is the left channel, and a rising edge transition indicates that the next data transferred is the right channel.

a. Not always available, refer to *Section 48.3: SPI implementation* to check if 24 and 32-bit data widths are supported.

Figure 980. Master I2S Philips protocol waveforms (16/32-bit full accuracy)



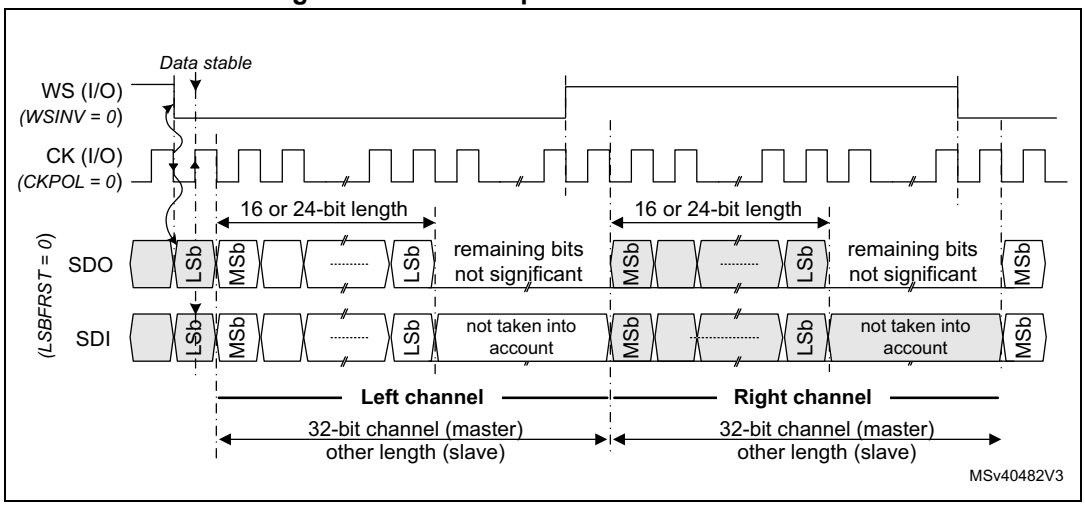
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 48.3: SPI implementation* for the supported data sizes.

CKPOL is cleared to match the I2S Philips protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

Figure 980 shows an example of waveform generated by the SPI/I2S in the case where the channel length is equal to the data length. More precisely, this is true when CHLEN = 0 and DATLEN = 0b00 or when CHLEN = 1 and DATLEN = 0b10.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

Figure 981. I2S Philips standard waveforms



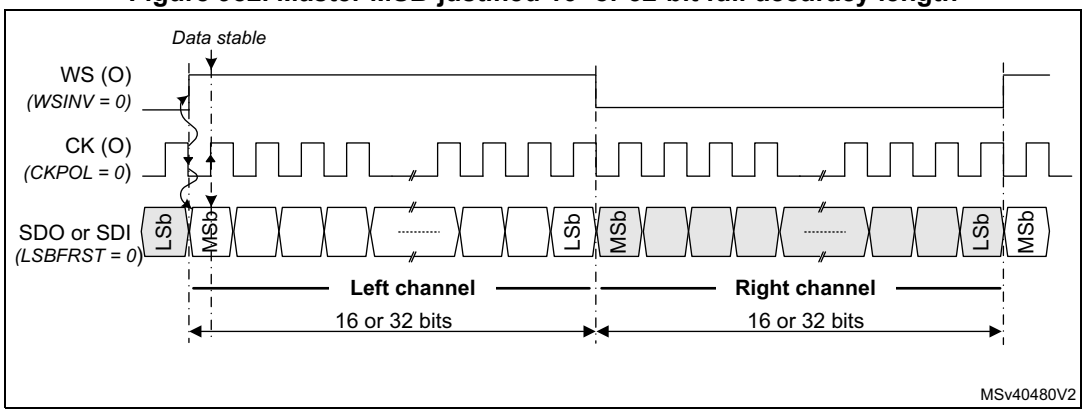
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 48.3: SPI implementation* for the supported data sizes.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in transmit mode. This is applicable for both master and slave mode.

MSB-justified standard

For this standard, the WS signal toggles when the first data bit is provided. The data transferred represents the left channel if WS is high and the right channel if WS is low.

Figure 982. Master MSB-justified 16- or 32-bit full-accuracy length

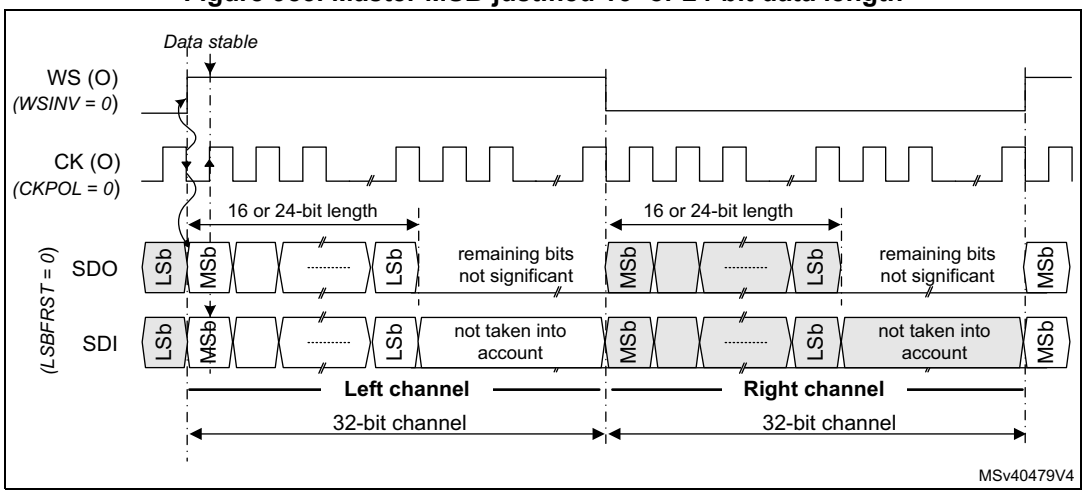


1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 48.3: SPI implementation* to check the supported data size.

CKPOL is cleared to match the I2S MSB-justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

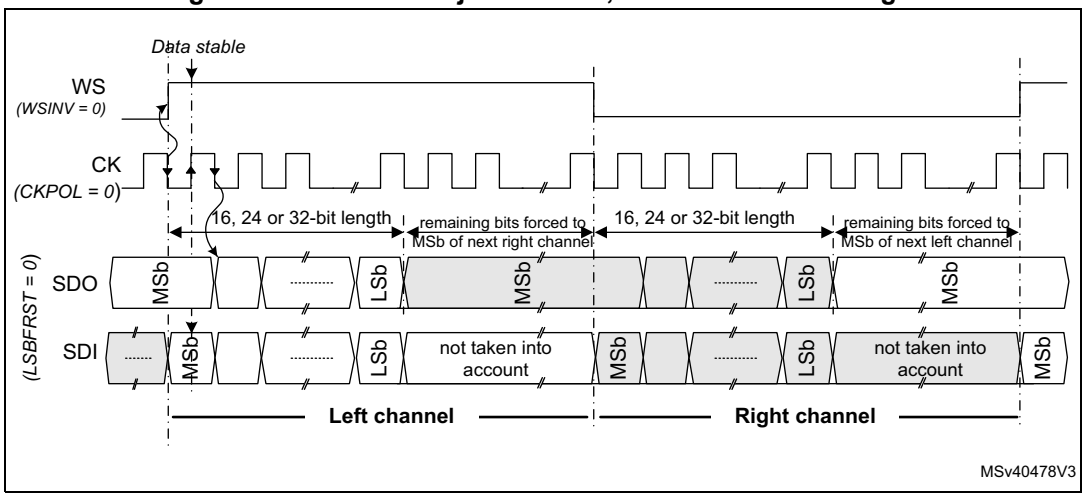
Figure 983. Master MSB-justified 16- or 24-bit data length



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 48.3: SPI implementation* to check the supported data size.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in master transmit mode. In slave transmit mode, the remaining bits are forced to the value of the first bit of the next data to be generated to avoid timing issues (see *Figure 984*).

Figure 984. Slave MSB-justified 16-, 24- or 32-bit data length



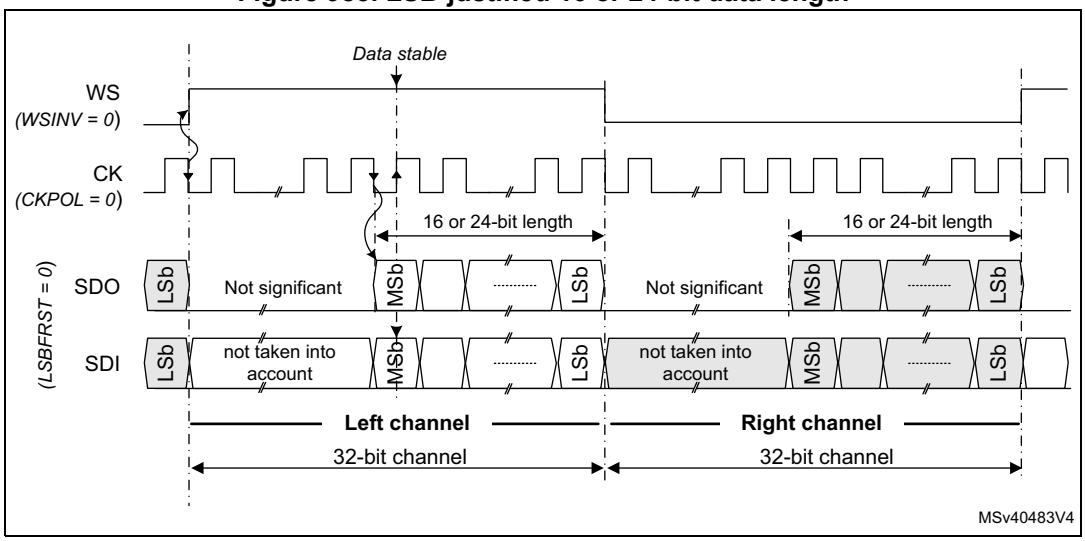
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 48.3: SPI implementation* to check the supported data size.

LSB-justified standard

This standard is similar to the MSB-justified standard in master mode (no difference for the 16- and 32-bit full-accuracy frame formats). The LSB-justified 16- or 32-bit full-accuracy format gives similar waveforms to MSB-justified mode (see *Figure 982*) because the channel and data have the same length.

Note: In the LSB-justified format, only 16- and 32-bit channel lengths are supported in master and slave mode. This is because it is not possible to transfer properly the data if the channel length is not known by the transmitter and receiver side.

Figure 985. LSB-justified 16 or 24-bit data length



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 48.3: SPI implementation* to check the supported data size.

CKPOL is cleared in order to match the I2S LSB-justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

PCM standard

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and can be selected using the PCMSYNC bit of SPI_I2SCFGR register.

In PCM long frame:

- The assertion time of the WS signal is fixed to 13 cycles of CK in master mode,
- The first data bit is received or transmitted as soon as the WS signal is asserted.

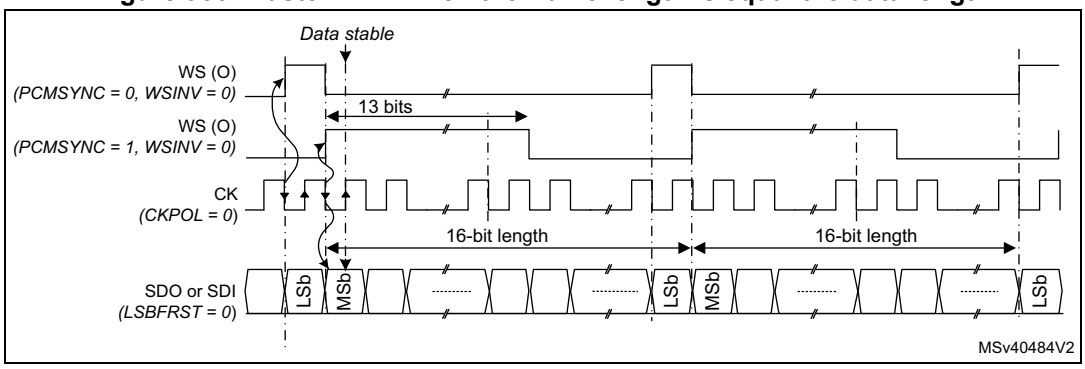
In PCM short frame:

- The assertion time of the WS signal is fixed to one cycle of CK in master mode,
- The first data bit is received or transmitted one cycle of CK after the WS assertion.

For both PCM modes:

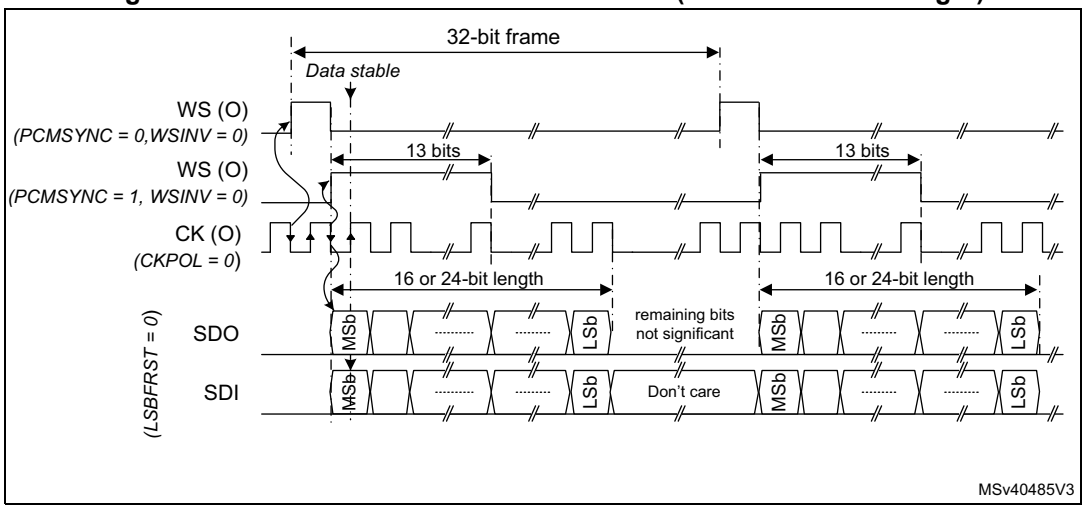
- The first data bit is MSb or LSb depending on the LSBFIRST bit value.
- The CK sampling edge can be selected thanks to CKPOL bit.
- The WS signal can be inverted thanks to the WSINV bit. See *Control of the WS inversion* for information concerning the handling of WS signal.

Figure 986. Master PCM when the frame length is equal the data length



A data size of 16 or 24 bits can be used when the channel length is set to 32 bits.

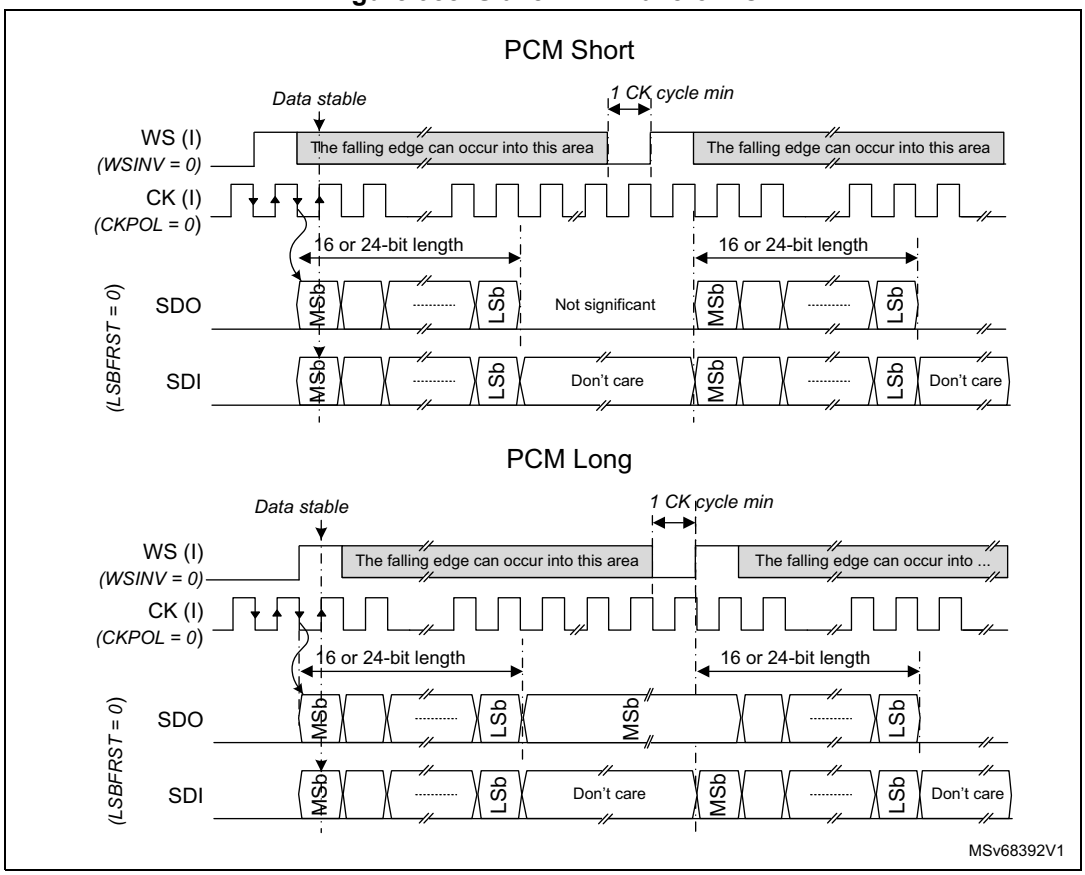
Figure 987. Master PCM standard waveforms (16 or 24-bit data length)



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 48.3: SPI implementation* to check the supported data size.

If the PCM protocol is used in slave mode, frame lengths can be different from 16 or 32 bits. As shown in *Figure 988*, in slave mode various pulse widths of WS can be accepted as the start of frame is detected by a rising edge of WS. The only constraint is that the WS must go back to its inactive state for at least one CK cycle.

Figure 988. Slave PCM waveforms



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 48.3: SPI implementation* to check the supported data size.

Note: In the case where the channel length is bigger than the data length, in slave PCM long, the transmission of the remaining bits is forced to the value of the first bit of the next data to be generated if the TXFIFO contains the next data to be transmitted.

In slave mode, CHLEN must be always programmed properly (whatever FIXCH value). For example, if CHLEN is cleared (16-bit length), the data transfer is truncated to 16 bits even if DATLEN is different from 0. To avoid this situation, CHLEN must be set, if DATLEN = 1 or 2.

48.9.6 Additional serial interface flexibility

Variable frame length in slave

In slave mode, channel lengths different from 16 or 32 bits can be accepted, as long as the channel length is bigger than the data length. This is true for all protocols except for the I2S LSB-justified protocol.

Data ordering

For all data formats and communication standards, it is possible to select the data ordering (MSb or LSb first) thanks to the bit LSBFRST located into *SPI/I2S configuration register 2 (SPI_CFG2)*.

Selection of the CK sampling edge

The CKPOL bit located into *SPI/I2S configuration register (SPI_I2SCFGR)* allows the user to choose the sampling edge polarity of the CK for slave and master modes, for all protocols.

- When CKPOL = 0, serial data SDO and WS (when master) are changed on the falling edge of CK and the serial data SDI and WS (when slave) are read on the rising edge.
- When CKPOL = 1, serial data SDO and WS (when master) are changed on the rising edge of CK and the serial data SDI and WS (when slave) are read on the falling edge.

Control of the WS inversion

It is possible to invert the default WS signal polarity for master and slave modes, for all protocols, by setting WSINV. By default the WS polarity is the following:

- In I2S Philips standard, WS is LOW for left channel, and HIGH for right channel
- In MSB/LSB-justified mode, WS is HIGH for left channel, and LOW for right channel
- In PCM mode, the start of frame is indicated by a rising edge of WS.

When WSINV is set, the WS polarity is inverted, then:

- In I2S Philips standard, WS is HIGH for left channel, and LOW for right channel
- In MSB/LSB-justified mode, WS is LOW for left channel, and HIGH for right channel
- In PCM mode, the start of frame is indicated by a falling edge of WS.

WSINV is located in the *SPI/I2S configuration register (SPI_I2SCFGR)*.

Control of the I/Os

The SPI/I2S block allows the settling of the WS and CK signals to their inactive state before enabling the SPI/I2S thanks to the AFCNTR bit of *SPI/I2S configuration register 2 (SPI_CFG2)*.

This can be done by programming CKPOL and WSINV using the following sequence:

Assuming that AFCNTR is initially cleared:

- Set I2SMOD = 1, (to inform the hardware that the CK and WS polarity is controlled via CKPOL and WSINV).
- Set bits CKPOL and WSINV to the wanted value.
- Set AFCNTR = 1.
  Then the inactive level of CK and WS I/Os is set according to CKPOL and WSINV values, even if the SPI/I2S is not yet enabled.
- Then performs the activation sequence of the I2S/PCM

Table 559 shows the level of WS and CK signals, when the AFCNTR bit is set, and before the SPI/I2S block is enabled (that is inactive level). Note that the level of WS also depends on the protocol selected.

Table 559. WS and CK level before SPI/I2S is enabled when AFCNTR = 1

WSINV	I2SSTD		WS level before SPI/I2S is enabled
0	I2S Std (00)	→	High
	Others	→	Low
1	I2S Std (00)	→	Low
	Others	→	High

CKPOL		CK level before SPI/I2S is enabled
0	→	Low
1	→	High

Note: The bit AFCNTR must not be set when the SPI is in slave mode.

Serial data line swapping

The direction of SDI and SDO depends on the IOSWP bit of *SPI/I2S configuration register 2 (SPI_CFG2)*, and on the slave/master mode. *Table 560* gives details on this feature.

Table 560. Serial data line swapping

Direction	IOSWP	Master mode		Slave mode	
		Input line	Output line	Input line	Output line
RX	0	SDI	-	SDO	-
	1	SDO	-	SDI	-
TX	0	-	SDO	-	SDI
	1	-	SDI	-	SDO
Full-duplex	0	SDI	SDO	SDO	SDI
	1	SDO	SDI	SDI	SDO

48.9.7 Startup sequence

When the bit SPE is cleared, the user is not allowed to read and write from/to the SPI_RXDR and SPI_TXDR registers, but the access to other registers is allowed.

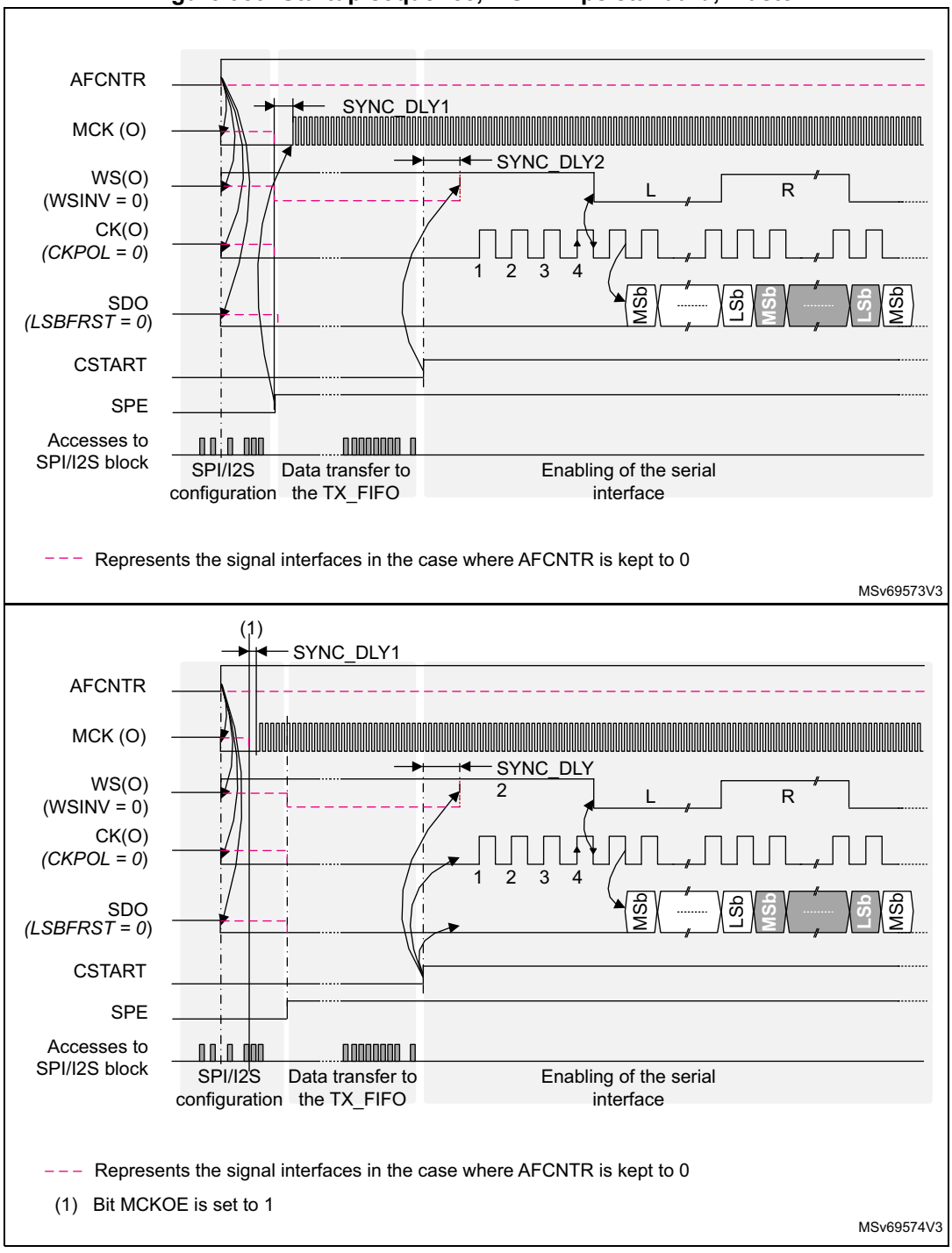
When the application wants to use the SPI/I2S block the user has to proceed as follows:

1. Ensure that the SPE is cleared, otherwise write SPE to 0.
2. Program all the configuration and control registers according to the wanted configuration. Refer to *Section 48.9.16* for detailed programming examples.
3. Clear all the status flags by setting the USPC, TIFREC, OVRC, and UDRC bits of the SPI_IFCR register. Note that if the flag SUSP is not cleared (via the SUSPC bit) the CSTART control bit has no effect.
4. Set the SPE bit to activate the SPI/I2S block. When this bit is set, the serial interface is still disabled, but the DMA and interrupt services are working, allowing for example, the data transfer into the TxFIFO. The generation of MCK can also be started when SPE goes to 1.
5. Set bit CSTART to activate the serial interface.

As shown in *Figure 989*, in I2S Philips standard master TX, the generation of the WS and CK signals starts after a resynchronization delay (SYNC_DLY2) when CSTART goes to 1

and the TxFIFO is not empty. Note that the CK bit clock is activated four rising edges before the falling edge of WS in order to ensure that the external slave device can detect properly WS transition. Other standards behave similarly.

Figure 989. Startup sequence, I2S Philips standard, master



1. As shown in the figure, MCK can be enabled as soon as the bit SPE is set. It is generated after a synchronization delay (SYNC_DLY1). See MCK generation in *Section 48.9.9: Clock generator* for more information.
2. When MCKOE is set, the MCK is generated after a synchronization delay (SYNC_DLY1). For more

information, see MCK generation in *Section 48.9.9: Clock generator*.

3. Note that the level of WS and CK signals are controlled by the SPI/I2S block during the configuration phase as soon as the AFCNTR bit is set.

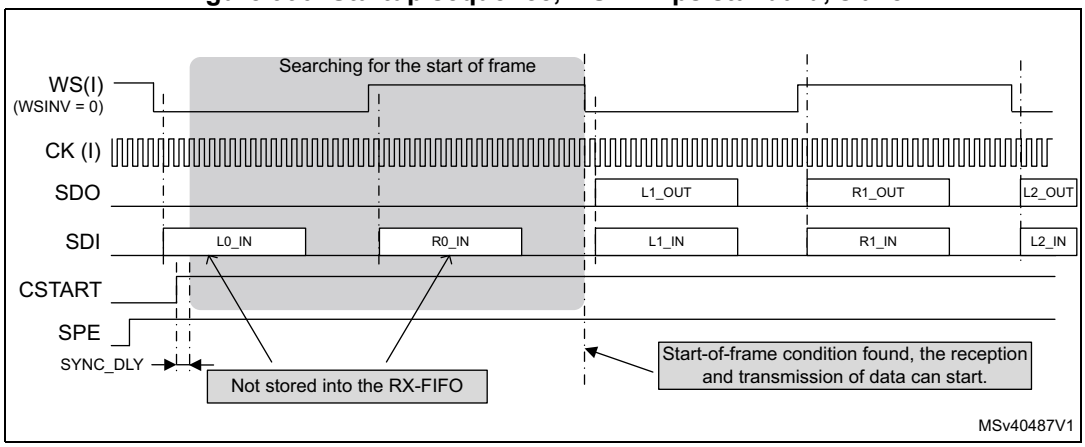
Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after about three periods of CK clock (SYNC_DLY2).

In slave mode, once the bit CSTART is set, the data transfer starts when the start-of-frame condition is met:

- For the I2S Philips standard, the start-of-frame condition is a falling edge of the WS signal. The transmission/reception starts one bit clock later.
  If WSINV = 1, then the start-of-frame condition is a rising edge.
- For other protocols, the start-of-frame condition is a rising edge of the WS signal. The transmission/reception starts at the rising edge of WS for MSB-aligned protocol. The transmission/reception starts one bit clock later for PCM protocol.
  If WSINV = 1, then the start-of-frame condition is a falling edge.

Figure 990 shows an example of startup sequence in I2S Philips standard, slave mode.

Figure 990. Startup sequence, I2S Philips standard, slave



Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after two periods of CK clock (SYNC_DLY).

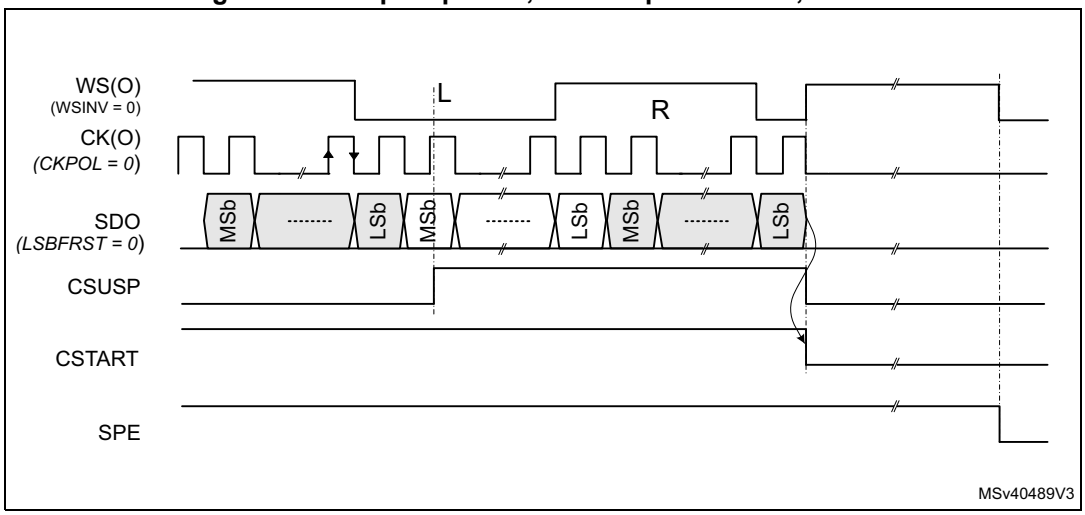
48.9.8 Stop sequence

The application can stop the I2S/PCM transfers by clearing the SPE bit. In that case the communication is stopped immediately, without waiting for the end of the current frame.

In master mode it is also possible to stop the I2S/PCM transfers at the end of the current frame. For that purpose, the user has to set the bit CSUSP, and polls the CSTART bit until it goes to 0. The CSTART bit goes to 0 when the current stereo (if an I2S mode was selected) or mono sample are completely shifted in or out. Then the SPE bit can be cleared.

Figure 991 shows an example of stop sequence in the case of master mode. The CSUSP bit is set during the transmission of the left sample, the transfer continue until the last bit of the right sample is transferred. Then CSTART and CSUSP go back to 0, CK and WS signals go back to their inactive state, and the user can clear the SPE bit.

Figure 991. Stop sequence, I2S Philips standard, master

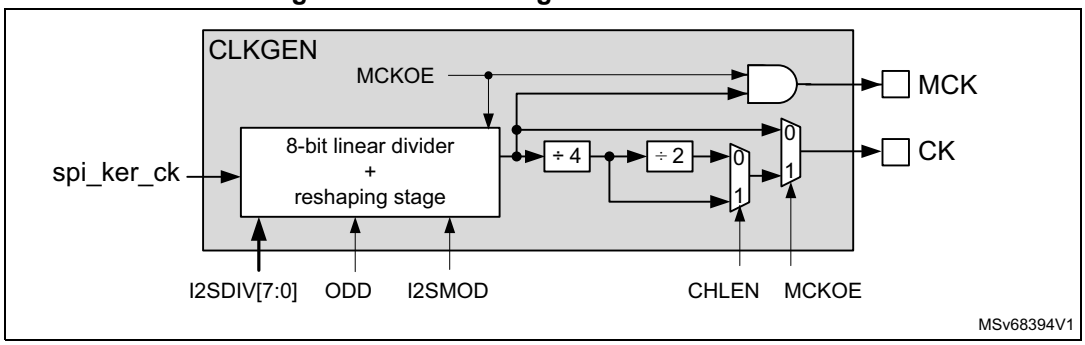


Note: In slave mode, the stop sequence is only controlled by the SPE bit.

48.9.9 Clock generator

When the I2S or PCM is configured in master mode, the user needs to program the clock generator in order to produce the frame synchronization (WS), the bit clock (CK) and the master clock (MCK) at the desired frequency.
If the I2S or PCM is used in slave mode, there is no need to configure the clock generator.

Figure 992. I²S clock generator architecture



The frequency generated on MCK, CK and WS depends mainly on I2SDIV, ODD, CHLEN, and MCKOE. The bit MCKOE indicates if a master clock need to be generated or not. The master clock has a frequency 256 or 128 times higher than the frame synchronization. This master clock is often required to provide a reference clock to external audio codecs.

Note: In master mode, there are no specific constraints on the ratio between the bus clock rate (F_{pclk}) and the bit clock (F_{CK}). The bus clock frequency must be high enough in order to support the data throughput.

When the master clock is generated (MCKOE = 1), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s_clk}}{256 \times \{(2 \times I2SDIV) + ODD\}}$$

And by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s_clk}}{128 \times \{(2 \times I2SDIV) + ODD\}}$$

In addition, the frequency of the MCK (F_{MCK}) is given by the formula:

$$F_{MCK} = \frac{F_{i2s_clk}}{\{(2 \times I2SDIV) + ODD\}}$$

When the master clock is disabled (MCKOE = 0), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s_clk}}{32 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

And by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s_clk}}{16 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

Where F_{WS} is the frequency of the frame synchronization, and F_{i2s_clk} is the frequency of the kernel clock provided to the SPI/I2S block.

Note: CHLEN and ODD can be either 0 or 1.

I2SDIV can take any values from 0 to 255 when ODD = 0, but when ODD = 1, the value I2SDIV = 1 is not allowed.

When I2SDIV = 0, then {(2 x I2SDIV) + ODD} is forced to 1.

Note: When {(2 x I2SDIV) + ODD} is odd, the duty cycle of the MCK or the CK signal is not 50%. Care must be taken when an odd ratio is used: it can impact margin on setup and hold time. For example if {(2 x I2SDIV) + ODD} = 5, then the duty cycle can be 40%.

Note: MCK duty cycle is 50 %, even for odd division ratios, when the kernel clock duty cycle is 50 %. When the division ratio is set (I2SDIV = 0), the MCK duty cycle is the same as the duty cycle of the kernel clock.

Table 561 provides examples of clock generator programming for I2S modes.

MCK generation

The master clock MCK is generated when the following conditions are met:

- I2SMOD must be equal to 1,
- I2SCFG must select a master mode,
- MCKOE must be set,
- SPE must be set

Table 561. CLKGEN programming examples for usual I2S frequencies

i2s_clk (MHz)	Channel length (bits)	I2SDIV	ODD	MCK	Sampling rate: F_{WS} (kHz)
12.288	16	12	0	No	16
12.288	32	6	0		16
12.288	16	6	0		32
12.288	32	3	0		32
49.152	16	16	0		48
49.152	32	8	0		48
49.152	16	8	0		96
49.152	32	4	0		96
49.152	16	4	0		192
49.152	32	2	0		192
4.096	16 or 32	0	-	Yes	16
24.576	16 or 32	3	0		
49.152	16 or 32	3	0		32
12.288	16 or 32	0	-		48
49.152	16 or 32	2	0		
61.44	16 or 32	2	1		
98.304	16 or 32	2	0		96
196.608	16 or 32	2	0		192

48.9.10 Internal FIFOs

The I2S interface can use a dedicated FIFO for the RX and the TX path. The samples to transmit can be written into the TxFIFO via the SPI_TXDR register. The reading of RxFIFO is performed via the SPI_RXDR register.

Data alignment and ordering

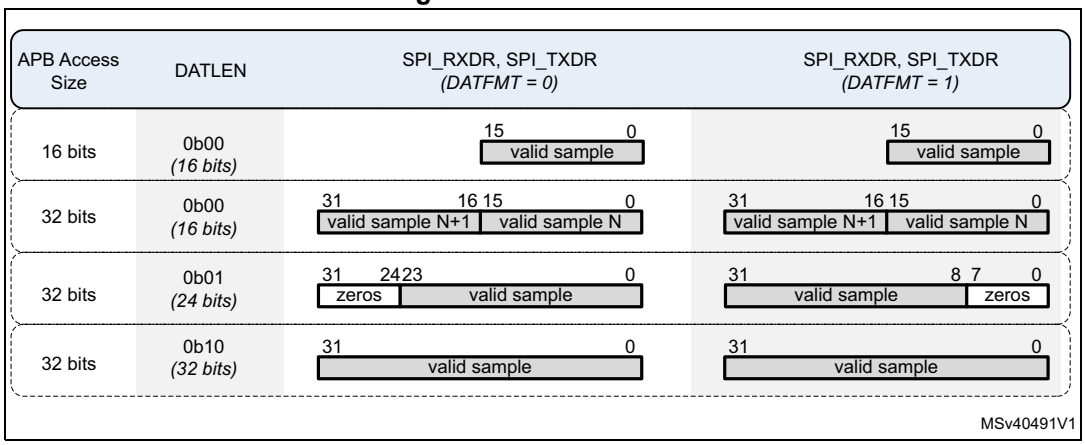
It is possible to select the data alignment into the SPI_RXDR and SPI_TXDR registers thanks to the DATFMT bit.

Note as well that the format of the data located into the SPI_RXDR or SPI_TXDR depends as well on the way those registers are accessed via the APB bus.

Figure 993 shows the allowed settings between APB access sizes, DATFMT, and DATLEN.

Note: Caution must be taken when the APB access size is 32 bits, and DATLEN = 0. For read operation the RxFIFO must contain at least two data, otherwise the read data are invalid. In the same way, for write operation, the TxFIFO must have at least two empty locations, otherwise a data can be lost.

Figure 993. Data Format



1. In I2S mode, the sample N represents the left sample, and the sample N+1 is the right sample.
2. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 48.3: SPI implementation* to check the supported data size.

It is possible to generate an interrupt or a DMA request according to a programmable FIFO threshold levels. The FIFO threshold is common to RX and TxFIFOs can be adjusted via FTHLV.

In I2S mode, the left and right audio samples are interleaved into the FIFOs. It means that for transmit operations, the user has to start to fill-up the TxFIFO with a left sample, followed by a right sample, and so on. For receive mode, the first data read from the RxFIFO is supposed to represent a left channel, the next one is a right channel, and so on.

Note that the read and write pointers of the FIFOs are reset when the bit SPE is cleared.

Refer to *Section 48.9.11* and *Section 48.9.15* for additional information.

FIFO size optimization

The basic element of the FIFO is the byte. This allows an optimization of the FIFO locations. For example when the data size is fixed to 24 bits, each audio sample takes three basic FIFO elements.

For example, a FIFO with 16 basic elements can have a depth of:

- 8 samples, if the DATLEN = 0 (16 bits),
- 5 samples, if the DATLEN = 1 (24 bits)[(a)],
- 4 samples, if the DATLEN = 2 (32 bits)[(a)].

48.9.11 FIFO status flags

Two status flags are provided for the application to fully monitor the state of the I2S interface. Both flags can generate an interrupt request. The receive interrupt is generated if RXPIE bit is enabled, the transmit interrupt is generated if TXPIE bit is enabled. Those bits are located into the SPI_IER register.

a. Not always available, refer to *Section 48.3: SPI implementation* in order to check if 24 and 32-bit data widths are supported.

TxFIFO threshold reached (TXP)

When set, this flag indicates that the TxFIFO contains at least FTHLV empty locations. Thus, FTHLV new data to be transmitted can be written to SPI_TXDR. The TXP flag is reset when the number of empty locations is lower than FTHLV. Note that TXP = 1, when the I2S is disabled (SPE bit is reset).

RxFIFO threshold reached (RXP)

When set, this flag indicates that there is at least FTHLV valid data into the RxFIFO, thus the user can read those data via SPI_RXDR. It is reset when the RxFIFO contains less than FTHLV data.

See *Section 48.10* for additional information on the interrupt function in I2S mode.

48.9.12 Handling of underrun situation

In transmit mode, an underrun situation is detected when a new data needs to be loaded into the shift register while the TxFIFO is already empty. In such a situation the UDR flag is set, and at least one audio frame contains unexpected data.

For I2S modes, there is a hardware mechanism to prevent misalignment situations (left and right channel swapped). When an underrun situation is detected, the last valid data present in the shift register is repeated if the real channel length matches the length selected by the CHLEN bit. In the other cases, an undefined data is repeated. Typically, if the block is programmed in slave TX mode, and the external master audio device is using channel lengths other than 16 or 32 bits, the repeated data value is undefined in case of underrun.

The following figure shows the case where an underrun occurs and the peripheral replays the last valid data on left and right channels as long as conditions of restart are not met. The transmission restarts:

- When there is enough data into the TxFIFO, and
- When the UDR flag is cleared by the software, and
- When the restart condition is met:
  - If the underrun occurs when a right channel data needs to be transmitted, the transmission restarts when a right channel needs to be transmitted, or
  - If the underrun occurs when a left channel data needs to be transmitted, the transmission restarts when a left channel needs to be transmitted.

Figure 994. Handling of underrun situation

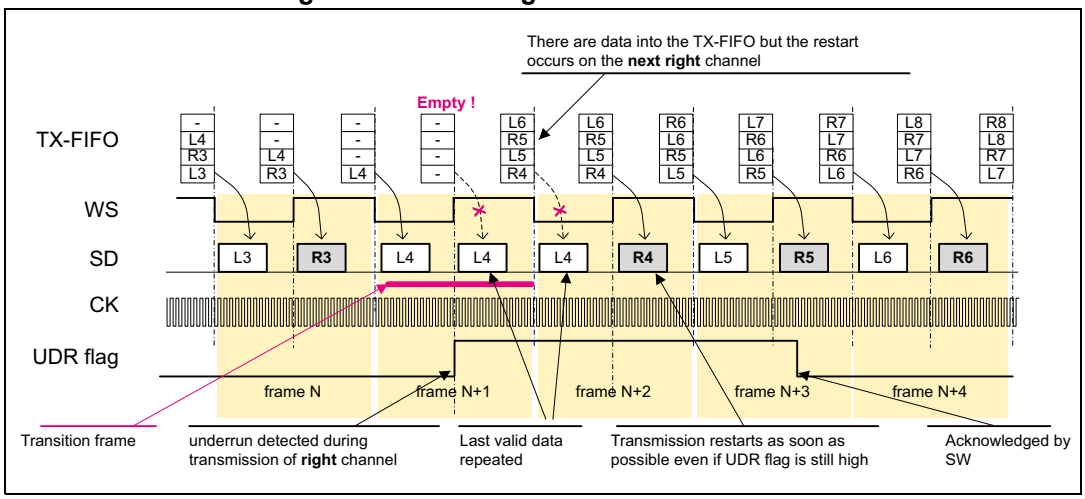
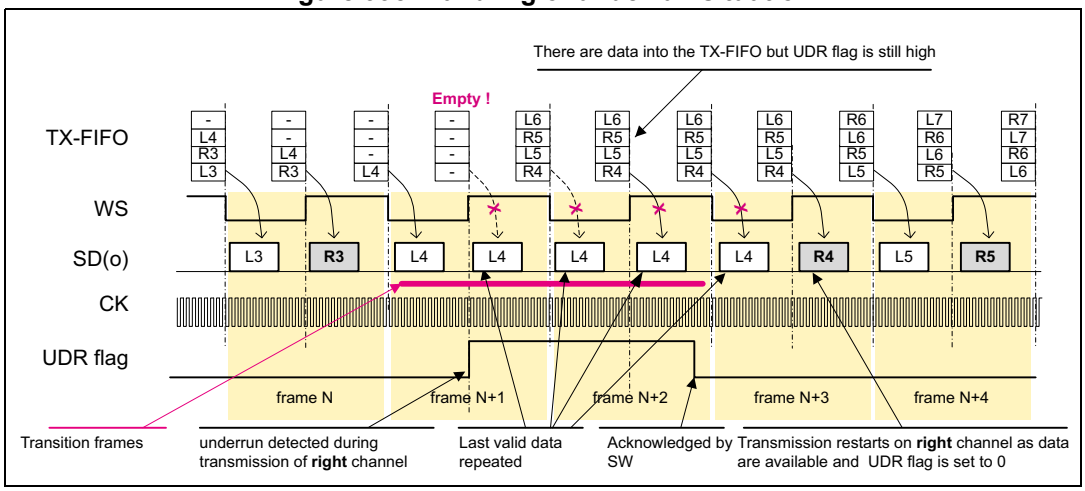


Figure 995. Handling of underrun situation



When the block is configured in one of the PCM modes, the transmission restarts at the start of the next frame, when there is enough data in the TxFIFO, and the UDR flag is cleared.

The UDR flag can trigger an interrupt if the UDRIE bit of the SPI_IER register is set. The UDR bit is cleared by setting the UDRC bit of the SPI_IFCR register.

Note: An underrun situation can occur in master or slave mode. In master mode, when an underrun occurs, the WS, CK, and MCK signals are not gated.

Due to resynchronization, any change of the UDR flag is taken into account by the hardware after at least two periods of CK clock.

48.9.13 Handling of overrun situation

In receive mode, an overrun situation is detected when the shift register needs to store a new data into the RxFIFO, while the RxFIFO is full. In such a situation the OVR flag is set, and the incoming data is lost.

In I2S mode, there is a hardware mechanism in to prevent misalignment situations (left and right channel swapped). As shown in the following figure, when an overrun occurs, the peripheral stops writing data into the RxFIFO as long as restart conditions are not met.

The reception restarts when there is enough room into the RxFIFO, and the OVR flag is cleared. The block starts by writing next the right channel into the RxFIFO if the overrun occurs when a right channel data is received or by writing the next left channel if the overrun occurs when a left channel data is received.

Figure 996. Handling of overrun situation

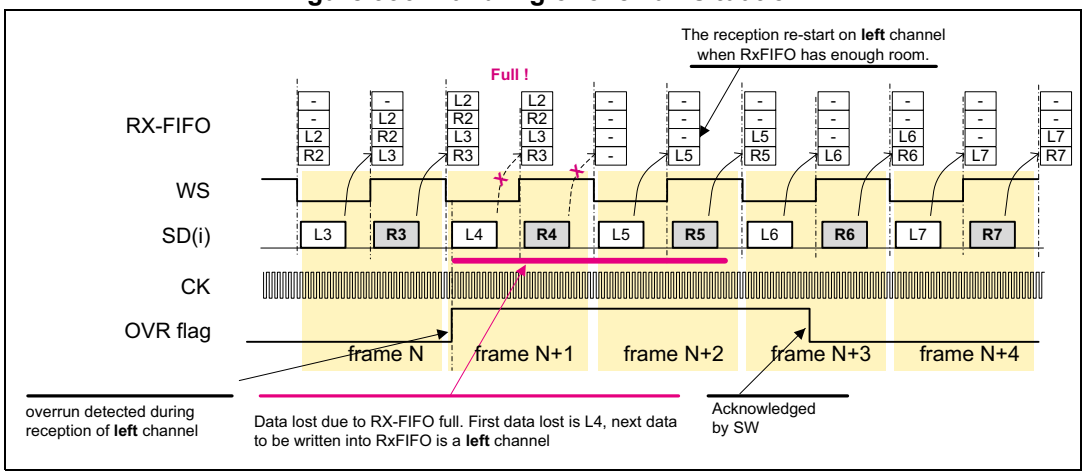
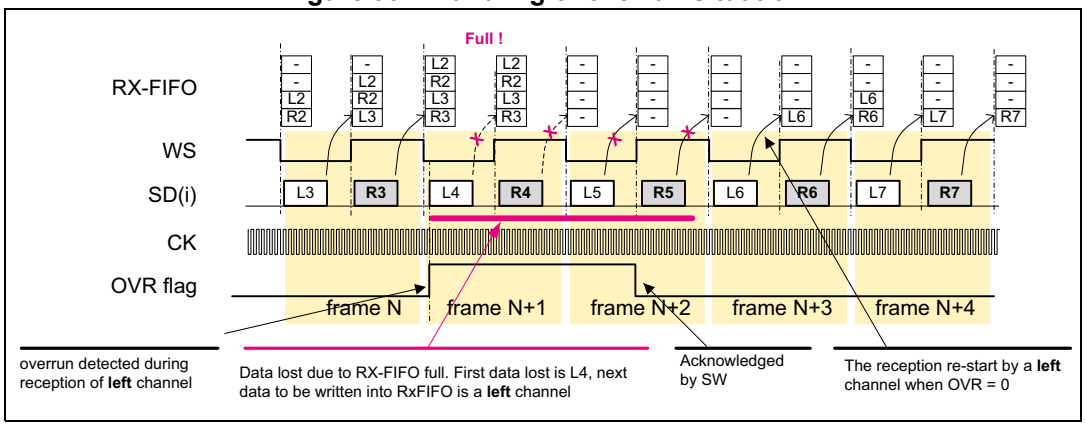


Figure 997. Handling of overrun situation



When the block is configured in PCM mode, after an overrun error, the block stops writing data in the RxFIFO as long as conditions of restart are not met. When there is enough room in the RxFIFO, and the OVR flag is cleared, the next received data are written in the RxFIFO.

An interrupt can be generated if the OVRIE bit of the SPI_IER register is set. The OVR bit is cleared by setting the OVRC bit of the SPI_IFCR register.

Note: An overrun situation can occur in master or slave mode. In master mode when an overrun occurs, the WS, CK, and MCK signals are not gated.

48.9.14 Frame error detection

When configured in slave mode, the SPI/I2S block detects two kinds of frame errors:

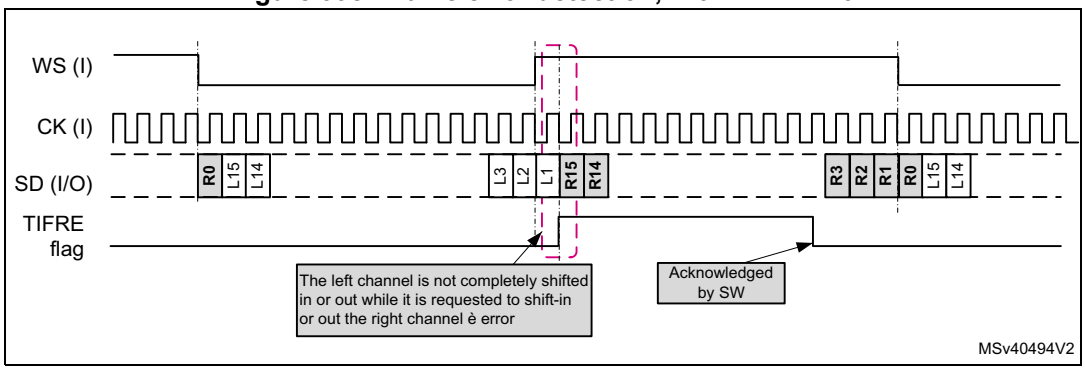
- A frame synchronization is received while the shift-in or shift-out of the previous data is not complete (early frame error). This mode is selected with FIXCH = 0.
- A frame synchronization occurs at an unexpected position. This mode is selected with FIXCH = 1.

In slave mode, if the frame length provided by the external master device is different from 32 or 64 bits, the user has to clear FIXCH. As the SPI/I2S synchronize each transfer with the WS there is no misalignment risk, but in a noisy environment, if a glitch occurs in the CK signal, a sample may be affected and the application is not aware of this.

If the frame length provided by the external master device is equal to 32 or 64 bits, then the user can set FIXCH and adjust accordingly CHLEN. As the SPI/I2S synchronize each transfer with the WS there is still no misalignment risk, and if the amount of bit clock between each channel boundary is different from CHLEN, the frame error flag (TIFRE) is set.

Figure 998 shows an example of frame error detection. The SPI/I2S block is in slave mode and the number of bit clock periods for the left channel are not enough to shift-in or shift-out the data. The figure shows that the ongoing transfer is interrupted and the next one is started in order to remain aligned to the WS signal.

Figure 998. Frame error detection, with FIXCH = 0

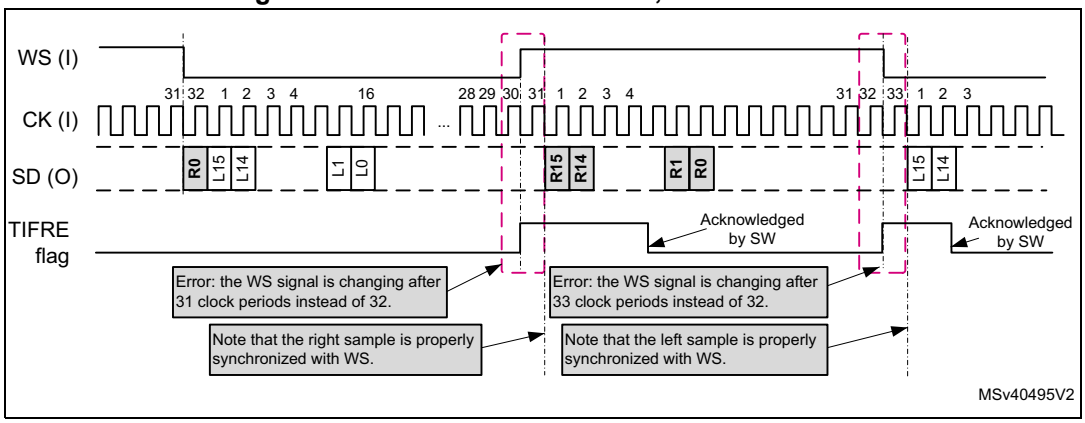


An interrupt can be generated if the TIFREIE bit is set. The frame error flag (TIFRE) is cleared by writing the TIFREC bit of the SPI_IFCR register to 1.

It is possible to extend the coverage of the frame error flag by setting the bit FIXCH. When this bit is set, then the SPI/I2S is expecting fixed channel lengths in slave mode. This means that the expected channel length can be 16 or 32 bits, according to CHLEN. As shown in *Figure 999*, in this mode, the SPI/I2S block is able to detect if the WS signal is changing at the expected moment (too early or too late).

Note: *Figure 998* and *Figure 999* show the mechanism for the slave transmit mode, but this is also true for slave receive and slave full-duplex.

Figure 999. Frame error detection, with FIXCH = 1



The frame error detection can be generally due to noisy environment disturbing the good reception of WS or CK signals.

Note: The SPI/I2S is not able to recover properly if an overrun and an early frame occur within the same frame. In this case the user has to disable and re-enable the SPI/I2S.

48.9.15 DMA interface

The I2S/PCM mode shares the same DMA requests lines than the SPI function. There is a separated DMA channel for TX and RX paths. Each DMA channel can be enabled via RXDMAEN and TXDMAEN bits of the SPI_CFG1 register.

In receive mode, the DMA interface is working as follows:

1. The hardware evaluates the RxFIFO level,
2. If the RxFIFO contains at least FTHLV samples, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the RxFIFO contains less than FTHLV samples, no DMA request is generated, and the hardware loop to step 1

In transmit mode, the DMA interface is working as follows:

1. The hardware evaluates the TxFIFO level,
2. If the TxFIFO contains at least FTHLV empty locations, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the TxFIFO contains less than FTHLV empty locations, no DMA request is generated, and the hardware loop to step 1

48.9.16 Programing examples

Master I2S Philips standard, full-duplex

This example shows how to program the interface for supporting the I2S Philips standard protocol in master full-duplex mode, with a sampling rate of 48 kHz, using the master clock. The assumption has been taken that the SPI/I2S is receiving a kernel clock (i2s_clk) of 61.44 MHz from the clock controller of the circuit. In the example above we took the assumption that the external audio codec needs to be programmed, for example via an I2C

interface before starting the transfer. In addition, it is supposed that this external audio codec needs the MCK to accept I2C commands.

Procedure

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed.
2. Program the AFMUX in order to select the wanted I/Os. In the current example MCK, CK, WS, SDO, SDI are needed.
3. Program the clock generator to provide the MCK clock, and to have a frame synchronization rate at exactly 48 kHz. Set I2SDIV to 2, ODD to 1, and MCKOE to 1.
4. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, CHLEN = 1 (32 bits per channel) DATLEN = 1 (24 bits), I2SSTD = 0 (I2S Philips standard), I2SCFG = 5 (master full-duplex), I2SMOD = 1, for I2S/PCM mode.
5. Adjust the FIFO threshold by setting the wanted value into FTHLV. For example, if a threshold of two audio samples is required, FTHLV = 1.
6. If the application wishes to perform data transfer via DMA, set the bits TXDMAEN and RXDMAEN.
7. If needed, the application can then configure the external audio codec via an I2C interface.
8. Clear all interrupt enable fields located in the SPI_IER register.
9. Clear all status flags, by setting SUSPC, TIFREC, OVRC, UDRC of the SPI_IFCR register.
10. Set the SPE bit, then the MCK is generated. In the example presented here, the TxFIFO is not filled-up as soon as SPE = 1. The application can then configure the external audio codec via an I2C interface even if audio samples to transmit are not yet available. An *Alternative sequence* is proposed hereafter.
11. When the application wants to start the data transfer, set the SPE bit. Then:
    – If the data transfer uses DMA:
    a) Program the DMA peripheral: two channels, one for RX and one for TX.
    b) Initialize the memory buffer with valid audio samples for TX path.
    c) Enable the DMA channels.
    d) Enable interrupt events such as UDRIE and OVRIE if needed in order to detect transfer errors.
    e) Enable the DMA channel. If TXDMAEN was set, the TxFIFO is filled up.
    – If the data transfer is done via interrupt:
    Enable the interrupt events UDRIE, OVRIE, TXPIE, and RXPIE (by writing 1). An interrupt request is immediately activated allowing the interrupt handler to fill-up the TxFIFO.
12. Finally, the SPI/I2S serial interface can be enabled by setting the bit CSTART. CSTART bit is located into the SPI_CR1 register.

Alternative sequence

- Steps 1 to 8 are similar to the previous sequence.
- If the data transfer uses DMA:
  – Program the DMA peripheral: two channels, one for RX and one for TX

  - Initialize the memory buffer with valid audio samples for the TX path
  - Enable interrupt events such as UDRIE and OVRIE if needed in order to detect transfer errors.
  - Enable the DMA channels,
- If the data transfer is done via interrupt:
  - Enable the interrupt events UDRIE, OVRIE, TXPIE, and RXPIE (by writing 1). An interrupt request is immediately activated allowing the interrupt handler to fill-up the TxFIFO.
- Set SPE, as soon as this bit is set to one the following actions may happen:
  - If the interrupt generation is enabled, the SPI/I2S generates an interrupt request allowing the interrupt handler to fill-up the TxFIFO.
  - If the DMA transfer is enabled, the SPI/I2S generates DMA requests in order to fill-up the TxFIFO
  - The MCK is generated. The application can then configure the external audio codec via an I2C interface.
- Finally, the SPI/I2S serial interface can be enabled by setting the bit CSTART. CSTART bit is located into the SPI_CR1 register.

Stop procedure in master mode

1. Set the bit CSUSP, in order to stop ongoing transfers
2. Check the value of the CSTART bit until it goes to 0
3. Clear the SUSP flag by setting SUSPC
4. Stop DMA peripheral, bus clock...
5. Clear bit SPE in order to disable the SPI/I2S block

Slave I2S Philips standard, receive

This example shows how to program the interface for supporting the I2S Philips standard protocol in slave receiver mode, with a sampling rate of 48 kHz. Note that in slave mode the SPI/I2S block cannot control the sample rate of the received samples. In this example we took the assumption that the external master device is delivering an I2S frame structure with a channel length of 24 bits. So, we cannot use the capability offered for frame error detection when FIXCH is set.

Procedure

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed,
2. Program the AFMUX in order to select the wanted I/Os. In the current example CK, WS, SDI,
3. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, FIXCH = 0 (because channel length is different from 16 and 32 bits), DATLEN = 0 (16 bits), I2SSTD = 0 (Philips protocol), I2SCFG = 1 (slave RX), I2SMOD = 1, for I2S mode.
4. Adjust the FIFO threshold by setting the wanted value into FTHLV. For example if a threshold of two audio samples is required, FTHLV = 1.
5. Clear all status flag registers.
6. Enable the flags that generate an interrupt such as OVRIE and TIFRE.
7. If the data transfer uses DMA:

– Program the DMA peripheral: one RX channel
– Enable the DMA channel,
– In the SPI/I2S block, enable the DMA by setting the RXDMAEN bit.

8. If the data transfer is done via interrupt, then the user has to enable the interrupt by setting the RXPIE bit.
9. Set SPE.
10. Finally, the user can set the bit CSTART in order to enable the serial interface. The SPI/I2S starts to store data into the RxFIFO on the next occurrence of left data transmitted by the external master device.

Stop procedure in slave mode

1. Clear bit SPE in order to disable the SPI/I2S block
2. Stop DMA peripheral, bus clock...

48.10 I2S interrupts

In PCM/I2S mode an interrupt (**spi_it**) or a wake-up event signal (**spi_wkup**) can be generated according to the events described in the *Table 562*.

Interrupt events can be enabled and disabled separately.

Table 562. I2S interrupt requests

Interrupt vector	Interrupt event	Event flag	Event/Interrupt clearing method	Exit Sleep mode	Exit Stop modes	Exit Standby mode
SPI	TxFIFO threshold reached	TXP	When the TxFIFO contains less than FTHLV empty locations	Yes	Yes	No
	RxFIFO threshold reached	RXP	When the RxFIFO contains less than FTHLV samples			
	Overrun error	OVR	Write OVRC to 1			
	Underrun error	UDR	Write UDRC to 1			
	Frame error flag	TIFRE	Write TIFREC to 1		No	

48.11 SPI/I2S registers

48.11.1 SPI/I2S control register 1 (SPI_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK
															rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
rw	rw	rw	rw	rw	w	rs	rw								rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **IOLOCK**: locking the AF configuration of associated I/Os

This bit can be changed by software only when SPI is disabled (SPE = 0). It is cleared by hardware if a MODF event occurs

When this bit is set, the SPI_CFG2 register content cannot be modified. This bit is write-protected when SPI is enabled (SPE = 1).

0: AF configuration is not locked

1: AF configuration is locked

Bit 15 **TCRCINI**: CRC calculation initialization pattern control for transmitter

0: all zero pattern is applied

1: all ones pattern is applied

Bit 14 **RCRCINI**: CRC calculation initialization pattern control for receiver

0: All zero pattern is applied

1: All ones pattern is applied

Bit 13 **CRC33_17**: Full size (33-bit or 17-bit) CRC polynomial configuration

0: Full size (33-bit or 17-bit) CRC polynomial is not used

1: Full size (33-bit or 17-bit) CRC polynomial is used

Bit 12 **SSI:** Internal slave select signal input level

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the peripheral NSS input internally and the I/O value of the NSS pin is ignored.

Bit 11 **HDDIR:** Rx/Tx direction at half-duplex mode

In half-duplex configuration the HDDIR bit establishes the Rx/Tx direction of the data transfer. This bit is ignored in Full-Duplex or any Simplex configuration.

0: SPI is receiver

1: SPI is transmitter

Bit 10 **CSUSP:** master suspend request

This bit reads as zero.

In master mode, when this bit is set by software, the CSTART bit is reset at the end of the current frame and communication is suspended. The user has to check SUSP flag to check end of the frame transfer.

The master mode communication must be suspended (using this bit or keeping TXDR empty) before going to Low-power mode.

After software suspension, SUSP flag must be cleared and SPI disabled and re-enabled before the next transfer starts.

Bit 9 **CSTART:** master transfer start

This bit can be set by software if SPI is enabled only to start an SPI or I2S/PCM communication. In SPI mode, it is cleared by hardware when end of transfer (EOT) flag is set or when a transfer suspend request is accepted. In I2S/PCM mode, it is also cleared by hardware as described in the *Section 48.9.8: Stop sequence*.

In SPI mode, the bit is taken into account at master mode only. If transmission is enabled, communication starts or continues only if any data is available in the transmission FIFO.

0: master transfer is at idle

1: master transfer is ongoing or temporary suspended by automatic suspend

Bit 8 **MASRX:** master automatic suspension in Receive mode

This bit is set and cleared by software to control continuous SPI transfer in master receiver mode and automatic management in order to avoid overrun condition.

When SPI communication is suspended by hardware automatically, it may happen that few bits of next frame are already clocked out due to internal synchronization delay.

This is why the automatic suspension feature is not reliable under one or both of the following conditions, where the master may still enter an overrun state and lose some received data:

– When data frames are configured to be shorter than 8 bits.

– When the SCK pad is exposed to a high external capacitive load.

This can cause a significant delay in synchronizing the internal SCK feedback signal, resulting in master-suspension latency that exceeds a single SCK bit period.

To prevent any overrun state, it is strongly recommended to set the MIDI parameter to a value equal to or greater than 4.

After software clearing of the SUSP bit, the communication resumes and continues by subsequent bits transfer without any next constraint. Before the SUSP bit is cleared, the user must release the RxFIFO space as much as possible by reading out all the data packets available at RxFIFO based on the RXP flag indication to prevent any subsequent suspension.

0: SPI flow/clock generation is continuous, regardless of overrun condition. (data are lost)

1: SPI flow is suspended temporary on RxFIFO full condition, before reaching overrun condition. The SUSP flag is set when the SPI communication is suspended.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **SPE:** serial peripheral enable

This bit is set by and cleared by software.

When SPE = 1, SPI data transfer is enabled, the SPI_CFG1 and SPI_CFG2 configuration registers, CRCPOLY, UDRDR, IOLOCK bit in the SPI_CR1 register are write protected. They can be changed only when SPE = 0.

When SPE = 0 any SPI operation is stopped and disabled, all the pending requests of the events with enabled interrupt are blocked except the MODF interrupt request (but their pending still propagates the request of the spi_plck clock), the NSS output is deactivated at master, the RDY signal keeps not ready status at slave, the internal state machine is reseted, all the FIFOs content is flushed, CRC calculation initialized, receive data register is read zero. SPE is cleared and cannot be set when MODF error flag is active.

0: Serial peripheral disabled.

1: Serial peripheral enabled

48.11.2 SPI/I2S control register 2 (SPI_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSIZE[15:0]**: number of data at current transfer

When these bits are changed by software, the SPI must be disabled.

Endless transfer is initialized when CSTART is set while zero value is stored in TSIZE. TSIZE cannot be set to 0xFFFF for a full-featured instance and 0x3FF for a limited-featured instance, when CRC is enabled.

Note: TSIZE[15:10] bits are reserved for limited-featured instances, and must be kept at reset value.

48.11.3 SPI/I2S configuration register 1 (SPI_CFG1)

Address offset: 0x008

Reset value: 0x0007 0007

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BPASS	MBR[2:0]			Res.	Res.	Res.	DRDS	Res.	CRCEN	Res.	CRCSIZE[4:0]				
rw	rw	rw	rw				rw		rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0]				DSIZE[4:0]				
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BPASS:** bypass of the prescaler at master baud rate clock generator
0: bypass is disabled
1: bypass is enabled

Bits 30:28 **MBR[2:0]:** master baud rate prescaler setting
000: SPI master clock/2
001: SPI master clock/4
010: SPI master clock/8
011: SPI master clock/16
100: SPI master clock/32
101: SPI master clock/64
110: SPI master clock/128
111: SPI master clock/256
Note: MBR setting is considered at slave working at TI mode, too (see Section 48.5.1: TI mode).

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **DRDS:** data sampling delay on master input (MISO)
This delay is used to compensate the delay on the MISO line (for example due to the signal optical isolation).
0: Sampling is performed on the configured SCK edge (no delay is applied)
1: Sampling is postponed by a fixed delay equal to a half SCK cycle (the sampling is performed just at the end of the bit validity interval at MISO line).
Note: The DRDS setting has no impact on the other SCK management nor on the data frame communication format. When the delay is applied together with CRC enabled, the CRC computation and evaluation is also delayed.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **CRCEN:** hardware CRC computation enable
0: CRC calculation disabled
1: CRC calculation enabled

Bit 21 Reserved, must be kept at reset value.

Bits 20:16 **CRCSIZE[4:0]:** length of CRC frame to be transferred and compared

Most significant bits are taken into account from polynomial calculation when CRC result is transferred or compared. The length of the polynomial is not affected by this setting.

The value must be equal or a multiple of the data size configured through the DSIZE bitfield. Its maximum size corresponds to the instance maximum data size.

00000: not used
00001: not used
00010: not used
00011: 4 bits
00100: 5 bits
00101: 6 bits
00110: 7 bits
00111: 8 bits
.....
11101: 30 bits
11110: 31 bits
11111: 32 bits

Note: The most significant bit at CRCSIZE bit field is reserved for the peripheral instances where data size is limited to 16 bits.

The CRCSIZE[2:0] bits are fixed to 1 for the peripheral instances with a limited set of features.

Bit 15 **TXDMAEN:** Tx DMA stream enable

0: Tx DMA disabled
1: Tx DMA enabled

Bit 14 **RXDMAEN:** Rx DMA stream enable

0: Rx-DMA disabled
1: Rx-DMA enabled

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **UDRCFG:** behavior of slave transmitter at underrun condition

For more details see *Figure 978: Optional configurations of the slave behavior when an underrun condition is detected*.

0: slave sends a constant pattern defined by the user in the SPI_UDRDR register

1: Slave repeats the last received data from master. When slave is configured at transmit only mode (COMM[1:0] = 01), all zeros pattern is repeated.

Bits 8:5 **FTHLV[3:0]:** FIFO threshold level

Defines number of data frames in a single data packet. It is recommended that the size of the packet does not exceed 1/2 of FIFO space.

SPI interface is more efficient if configured packet sizes are aligned with data register access parallelism:

–If SPI data register is accessed as a 16-bit register and DSIZE ≤ 8 bits, better to select FTHLV = 2, 4, 6.

–If SPI data register is accessed as a 32-bit register and DSIZE > 8 bits, better to select FTHLV = 2, 4, 6, while if DSIZE ≤ 8 bits, better to select FTHLV = 4, 8, 12.

0000: 1-data
0001: 2-data
0010: 3-data
0011: 4-data
0100: 5-data
0101: 6-data
0110: 7-data
0111: 8-data
1000: 9-data
1001: 10-data
1010: 11-data
1011: 12-data
1100: 13-data
1101: 14-data
1110: 15-data
1111: 16-data

Note: FTHLV[3:2] bits are reserved for instances with a limited set of features.

Bits 4:0 **DSIZE[4:0]:** number of bits in a single SPI data frame

00000: not used
00001: not used
00010: not used
00011: 4 bits
00100: 5 bits
00101: 6 bits
00110: 7 bits
00111: 8 bits
.....
11101: 30 bits
11110: 31 bits
11111: 32 bits

Note: The most significant bit DSIZE[4] is reserved for instances which data size is limited to 16 bits.

DSIZE[2:0] bits are fixed to 1 for instances with a limited set of features.

48.11.4 SPI/I2S configuration register 2 (SPI_CFG2)

Address offset: 0x00C

Reset value: 0x0000 0000

The content of this register is write-protected when SPI is enabled or the IOLOCK bit is set in the SPI_CR1 register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
rw	rw	rw						rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AFCNTR:** alternate function GPIOs control

This bit is taken into account when SPE = 0 only.

When SPI must be disabled temporary for a specific configuration reason (for example CRC reset, CPHA or HDDIR change) setting this bit prevents any glitches on the associated outputs configured at alternate function mode by keeping them forced at state corresponding the current SPI configuration.

0: The peripheral takes no control of GPIOs while it is disabled

1: The peripheral keeps always control of all associated GPIOs

Note: This bit can be also used in PCM and I2S modes.

The bit AFCNTR must not be set, when the block is in slave mode.

Bit 30 **SSOM**: NSS output management in master mode

This bit is taken into account in master mode when SSOE is enabled. It allows the NSS output to be configured between two consecutive data transfers.

0: NSS is kept at active level until data transfer is complete, it becomes inactive with EOT flag

1: SPI data frames are interleaved with NSS nonactive pulses when MIDI[3:0] > 1

Bit 29 **SSOE:** NSS output enable

This bit is taken into account in master mode only

0: NSS output is disabled and the SPI can work in multimaster configuration

1: NSS output is enabled. The SPI cannot work in a multimaster environment. It forces the NSS pin at inactive level after the transfer is complete or SPI is disabled with respect to SSOM, MIDI, MSSI, SSIOP bits setting

Bit 28 **SSIOP**: NSS input/output polarity

0: low level is active for NSS signal

1: high level is active for NSS signal

Bit 27 Reserved, must be kept at reset value.

Bit 26 **SSM:** Software management of internal slave select signal input

0: Input value of the internal slave select signal is determined by the external NSS hardware pin

1: Input value of the internal slave select signal is determined by the SSI bit controlled by software

Note: When master uses hardware NSS output (SSM = 0 and SSOE = 1), the internal slave select signal input is forced to not active state internally to prevent master mode fault error.

Bit 25 **CPOL:** clock polarity
0: SCK signal is at 0 when idle
1: SCK signal is at 1 when idle

Bit 24 **CPHA:** clock phase
0: the first clock transition is the first data capture edge
1: the second clock transition is the first data capture edge

Bit 23 **LSBFRST:** data frame format
0: MSB transmitted first
1: LSB transmitted first
Note: This bit can be also used in PCM and I2S modes.

Bit 22 **MASTER:** SPI master
0: SPI slave
1: SPI master

Bits 21:19 **SP[2:0]:** serial protocol
000: SPI Motorola
001: SPI TI
others: reserved, must not be used

Bits 18:17 **COMM[1:0]:** SPI Communication Mode
00: full-duplex
01: simplex transmitter
10: simplex receiver
11: half-duplex

Bit 16 Reserved, must be kept at reset value.

Bit 15 **IOSWP:** swap functionality of MISO and MOSI pins
When this bit is set, the function of MISO and MOSI pins alternate functions are inverted. Original MISO pin becomes MOSI and original MOSI pin becomes MISO.
0: no swap
1: MOSI and MISO are swapped
Note: This bit can be also used in PCM and I2S modes to swap SDO and SDI pins.

Bit 14 **RDIOP:** RDY signal input/output polarity
0: high level of the signal means the slave is ready for communication
1: low level of the signal means the slave is ready for communication

Bit 13 **RDIOM:** RDY signal input/output management
0: RDY signal is defined internally fixed as permanently active (RDIOP setting has no effect)
1: RDY signal is overtaken from alternate function input (at master case) or output (at slave case) of the dedicated pin (RDIOP setting takes effect)
Note: When DSIZE in the SPI_CFG1 register is configured shorter than 8 bits, the RDIOM bit must be kept at zero.

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:4 **MIDI[3:0]**: Master Inter-Data Idleness

Specifies minimum time delay (expressed in SPI clock cycles periods) inserted between two consecutive data frames in master mode.

0000: no delay

0001: 1 clock cycle period delay

...

1111: 15 clock cycle periods delay

Note: This feature is not supported in TI mode.

Bits 3:0 **MSSI[3:0]**: Master NSS Idleness

Specifies an extra delay, expressed in number of SPI clock cycle periods, inserted additionally between active edge of NSS opening a session and the beginning of the first data frame of the session in master mode when SSOE is enabled.

0000: no extra delay

0001: 1 clock cycle period delay added

...

1111: 15 clock cycle periods delay added

Note: This feature is not supported in TI mode.
To include the delay, the SPI must be disabled and re-enabled between sessions.

48.11.5 SPI/I2S interrupt enable register (SPI_IER)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 9 **MODFIE**: mode Fault interrupt enable

0: MODF interrupt disabled

1: MODF interrupt enabled

Bit 8 **TIFREIE**: TIFRE interrupt enable

0: TIFRE interrupt disabled

1: TIFRE interrupt enabled

Bit 7 **CRCEIE:** CRC error interrupt enable

0: CRC interrupt disabled

1: CRC interrupt enabled

Bit 6 **OVRIE:** OVR interrupt enable

0: OVR interrupt disabled

1: OVR interrupt enabled

Bit 5 **UDRIE:** UDR interrupt enable
0: UDR interrupt disabled
1: UDR interrupt enabled

Bit 4 **TXTFIE:** TXTF interrupt enable
0: TXTF interrupt disabled
1: TXTF interrupt enabled

Bit 3 **EOTIE:** EOT, SUSP and TXC interrupt enable
0: EOT/SUSP/TXC interrupt disabled
1: EOT/SUSP/TXC interrupt enabled

Bit 2 **DXPIE**: DXP interrupt enabled
DXPIE is set by software and cleared by TXTF flag set event.
0: DXP interrupt disabled
1: DXP interrupt enabled

Bit 1 **TXPIE:** TXP interrupt enable
TXPIE is set by software and cleared by TXTF flag set event.
0: TXP interrupt disabled
1: TXP interrupt enabled

Bit 0 **RXPIE:** RXP interrupt enable
0: RXP interrupt disabled
1: RXP interrupt enabled

48.11.6 SPI/I2S status register (SPI_SR)

Address offset: 0x014

Reset value: 0x0000 1002

All the flags of this register are not cleared automatically when the SPI is reenabled. They require specific clearing access exclusively via the flag clearing register as noted in the bits descriptions below.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DXP	TXP	RXP
r	r	r	r	r		r	r	r	r	r	r	r	r	r	r

Bits 31:16 **CTSIZE[15:0]**: number of data frames remaining in current TSIZE session
The value is not quite reliable when traffic is ongoing on bus .
Note: CTSIZE[15:0] bits are not available in instances with limited set of features.

Bit 15 **RXWNE:** RxFIFO word not empty
0: less than four bytes of RxFIFO space is occupied by data
1: at least four bytes of RxFIFO space is occupied by data
Note: This bit value does not depend on DSIZE setting and keeps together with RXPLVL[1:0] information about RxFIFO occupancy by residual data.

Bits 14:13 **RXPLVL[1:0]:** RxFIFO packing level

When RXWNE = 0 and data size is set up to 16 bits, the value gives number of remaining data frames persisting at RxFIFO.

When data size is greater than 16 bits, these bits are always read as 00. As a result, the single data frame received in the FIFO cannot be detected neither by RWNE nor by RXPLVL bits if the data size is set from 17 to 24 bits. The user must then apply other methods to detect the number of data received, such as monitor the EOT event when TSIZE > 0 or RXP events when FTHLV = 0.

00: no next frame is available at RxFIFO

01: 1 frame is available

10: 2 frames are available*

11: 3 frames are available*

Note: (*): Possible value when data size is set up to 8 bits only.

Bit 12 **TXC:** TxFIFO transmission complete

The flag behavior depends on TSIZE setting.

When TSIZE = 0, the TXC is changed by hardware exclusively and it raises each time the TxFIFO becomes empty and there is no activity on the bus.

If TSIZE ≠ 0 there is no specific reason to monitor TXC as it just copies the EOT flag value including its software clearing. The TXC generates an interrupt when EOTIE is set.

This flag is set when SPI is reset or disabled.

0: current data transfer is still ongoing, data is available in TxFIFO or last frame transmission is on going.

1: last TxFIFO frame transmission complete

Bit 11 **SUSP:** suspension status

In master mode, SUSP is set by hardware either as soon as the current frame is complete after CSUSP request is done or at master automatic suspend receive mode (the MASRX bit is set in the SPI_CR1 register) on RxFIFO full condition.

SUSP generates an interrupt when EOTIE is set.

This bit must be cleared prior to disabling the SPI. This is done by setting the SUSPC bit of SPI_IFCR exclusively.

0: SPI not suspended (master mode active or other mode).

1: master mode is suspended (current frame completed).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODF**: mode fault

When MODF is set, SPE and IOLOCK bits of the SPI_CR1 register are reset and setting SPE again is blocked until MODF is cleared.

This bit is cleared by writing 1 to the MODFC bit of the SPI_IFCR exclusively.

0: no mode fault

1: mode fault detected.

Bit 8 **TIFRE**: TI frame format error

This bit is cleared by writing 1 to the TIFREC bit of the SPI_IFCR exclusively.

0: no TI Frame Error

1: TI frame error detected

Bit 7 **CRCE:** CRC error

This bit is cleared when SPI is disabled or by writing 1 to the CRCEC bit of the SPI_IFCR.

0: no CRC error

1: CRC error detected

Bit 6 **OVR:** overrun

This bit is cleared when SPI is disabled or by writing 1 to the OVRC bit of the SPI_IFCR.

0: no overrun

1: overrun detected

Bit 5 **UDR:** underrun

This bit is cleared when SPI is disabled or by writing 1 to the UDRC bit of the SPI_IFCR.

0: no underrun

1: underrun detected

Note: In SPI mode, the UDR flag applies to slave mode only. In I2S/PCM mode, (when available) this flag applies to master and slave mode.

Bit 4 **TXTF:** transmission transfer filled

TXTF is set by hardware as soon as all of the data packets in a transfer have been submitted for transmission by application software or DMA, that is when TSIZE number of data have been pushed into the TxFIFO.

This bit is cleared by software write 1 to the TXTFC bit of the SPI_IFCR exclusively.

The TXTF flag triggers an interrupt if the TXTFIE bit is set.

TXTF setting clears the TXPIE and DXPIE masks so to off-load application software from calculating when to disable TXP and DXP interrupts.

0: upload of TxFIFO is ongoing or not started

1: TxFIFO upload is finished

Bit 3 **EOT:** end of transfer

EOT is set by hardware as soon as a full transfer is complete, that is when SPI is re-enabled or when TSIZE number of data have been transmitted and/or received on the SPI. EOT is cleared when SPI is re-enabled or by writing 1 to the EOTC bit of the SPI_IFCR register optionally.

EOT flag triggers an interrupt if the EOTIE bit is set.

If DXP flag is used until TXTF flag is set and DXPIE is cleared, EOT can be used to download the last packets contained into RxFIFO in one-shot.

In master, EOT event terminates the data transfer and handles NSS output optionally. When CRC is applied, the EOT event is extended over the CRC frame transfer.

To restart the internal state machine properly, SPI is strongly suggested to be disabled and re-enabled before next transfer starts despite its setting is not changed.

0: transfer is ongoing or not started

1: transfer complete

Bit 2 **DXP**: duplex packet

DXP flag is set whenever both TXP and RXP flags are set regardless SPI mode.

0: TxFIFO is Full and/or RxFIFO is Empty

1: both TxFIFO has space for write and RxFIFO contains for read a single packet at least

Bit 1 **TXP:** Tx-packet space available

TXP flag can be changed only by hardware. Its value depends on the physical size of the FIFO and its threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode and respective DATLEN[1:0] in I2S/PCM mode), and actual communication flow. If the data packet is stored by performing consecutive write operations to SPI_TXDR, TXP flag must be checked again once a complete data packet is stored at TxFIFO. TXP is set despite SPI TxFIFO becomes inaccessible when SPI is reset or disabled.

0: not enough free space at TxFIFO to host next data packet

1: enough free space at TxFIFO to host at least one data packet

Bit 0 **RXP:** Rx-packet available

The flag is changed by hardware. It monitors the total number of data currently available at RxFIFO if SPI is enabled. RXP value depends on the FIFO threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode and DATLEN[1:0] in I2S/PCM mode), and actual communication flow. If the data packet is read by performing consecutive read operations from SPI_RXDR, RXP flag must be checked again once a complete data packet is read out from RxFIFO.

0: RxFIFO is empty or an incomplete data packet is received

1: RxFIFO contains at least one data packet

48.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
				w		w	w	w	w	w	w	w			

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SUSPC:** Suspend flag clear

Writing a 1 into this bit clears SUSP flag of the SPI_SR register

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODFC**: mode fault flag clear

Writing a 1 into this bit clears MODF flag of the SPI_SR register

Bit 8 **TIFREC**: TI frame format error flag clear

Writing a 1 into this bit clears TIFRE flag of the SPI_SR register

Bit 7 **CRCEC:** CRC error flag clear

Writing a 1 into this bit clears CRCE flag of the SPI_SR register

Bit 6 **OVRC:** overrun flag clear

Writing a 1 into this bit clears OVR flag of the SPI_SR register

Bit 5 **UDRC:** underrun flag clear

Writing a 1 into this bit clears UDR flag of the SPI_SR register

Bit 4 **TXTFC:** transmission transfer filled flag clear

Writing a 1 into this bit clears TXTF flag of the SPI_SR register

Bit 3 **EOTC:** end of transfer flag clear

Writing a 1 into this bit clears EOT flag of the SPI_SR register

Bits 2:0 Reserved, must be kept at reset value.

48.11.8 SPI/I2S transmit data register (SPI_TXDR)

Address offset: 0x020

Reset value: 0x0000 0000

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

Bits 31:0 **TXDR[31:0]:** transmit data register

The register serves as an interface with TxFIFO. A write to it accesses TxFIFO.

Note: In SPI mode, data is always right-aligned. Alignment of data at I2S mode depends on DATLEN and DATFMT setting. Unused bits are ignored when writing to the register, and read as zero when the register is read.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is written by single access.

half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word-data can be written by single access.

word-wise (32 bit access). In this case 4 data-bytes or 2 half-word-data or word-data can be written by single access.

Write access of this register less than the configured data size is forbidden.

48.11.9 SPI/I2S receive data register (SPI_RXDR)

Address offset: 0x030

Reset value: 0x0000 0000

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

Bits 31:0 **RXDR[31:0]:** receive data register

The register serves as an interface with RxFIFO. When it is read, RxFIFO is accessed.

Note: In SPI mode, data is always right-aligned. Alignment of data at I2S mode depends on DATLEN and DATFMT setting. Unused bits are read as zero when the register is read. Writing to the register is ignored.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is read by single access

half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word data can be read by single access

word-wise (32 bit access). In this case 4 data-bytes or 2 half-word data or word-data can be read by single access.

Read access of this register less than the configured data size is forbidden.

48.11.10 SPI/I2S polynomial register (SPI_CRCPOLY)

Address offset: 0x040

Reset value: 0x0000 0107

The content of this register is write protected when SPI is enabled.

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

Bits 31:0 **CRCPOLY[31:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The default 9-bit polynomial setting 0x107 corresponds to default 8-bit setting of DSIZE. It is compatible with setting 0x07 used in other ST products with fixed length of the polynomial string, where the most significant bit of the string is always kept hidden.

Length of the polynomial is given by the most significant bit of the value stored in this register. It must be set greater than DSIZE. CRC33_17 bit must be set additionally with CRCPOLY register when DSIZE is configured to maximum data size and CRC is enabled (to keep polynomial length grater than data size).

Note: CRCPOLY[31:16] bits are reserved for instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

48.11.11 SPI/I2S transmitter CRC register (SPI_TXCRC)

Address offset: 0x044

Reset value: 0x0000 0000

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

Bits 31:0 **TXCRC[31:0]:** CRC register for transmitter

When CRC calculation is enabled, the TXCRC[31:0] bits contain the computed CRC value of the subsequently transmitted bytes. CRC calculation is initialized when the CRCEN bit of the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: This bitfield is not used in I2S mode.

Note: TXCRC[31-16] bits are reserved for instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of the CRCSIZE bitfield is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

48.11.12 SPI/I2S receiver CRC register (SPI_RXCRC)

Address offset: 0x048

Reset value: 0x0000 0000

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

Bits 31:0 **RXCRC[31:0]:** CRC register for receiver

When CRC calculation is enabled, the RXCRC[31:0] bits contain the computed CRC value of the subsequently received bytes. CRC calculation is initialized when the CRCEN bit of the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: This bitfield is not used in I2S mode.

Note: RXCRC[31-16] bits are reserved for the peripheral instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of CRCSIZE bit field is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

48.11.13 SPI/I2S underrun data register (SPI_UDRDR)

Address offset: 0x04C

Reset value: 0x0000 0000

The content of this register is write protected when SPI is enabled.

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

Bits 31:0 **UDRDR[31:0]:** data at slave underrun condition

The register is taken into account in slave mode and at underrun condition only. The number of bits considered depends on the DSIZE bit setting in the SPI_CFG1 register. Underrun condition handling depends on the UDRCFG bit setting in the SPI_CFG1 register.

Note: UDRDR[31-16] bits are reserved for the peripheral instances with data size limited to 16 bits. There is no constraint when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

48.11.14 SPI/I2S configuration register (SPI_I2SCFGR)

Address offset: 0x050

Reset value: 0x0000 0000

This register must be configured when the I2S is disabled (SPE = 0). The content of this register is not taken into account in SPI mode except for the I2SMOD bit, which needs to be kept at 0.

This register is reserved for instances not supporting I2S mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DATFMT	WSINV	FIXCH	CKPOL	CHLEN	DATLEN[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]			I2SMOD
	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

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

Bit 25 **MCKOE**: master clock output enable

0: Master clock output is disabled

1: Master clock output is enabled

Bit 24 **ODD**: odd factor for the prescaler

Refer to *Section 48.9.9: Clock generator* for details

0: Real divider value is = I2SDIV *2

1: Real divider value is = (I2SDIV * 2) + 1

Bits 23:16 **I2SDIV[7:0]**: I^2S linear prescaler
I2SDIV can take any values except the value 1, when ODD is also equal to 1.
Refer to *Section 48.9.9: Clock generator* for details

Bit 15 Reserved, must be kept at reset value.

Bit 14 **DATFMT**: data format
0: The data inside the SPI_RXDR or SPI_TXDR register are right aligned
1: The data inside the SPI_RXDR or SPI_TXDR register are left aligned.

Bit 13 **WSINV**: word select inversion
This bit is used to invert the default polarity of WS signal.
0: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS falling edge, and the right channel one CK cycle after the WS rising edge.
In MSB- or LSB-justified mode, the left channel is transferred when WS is HIGH, and the right channel when WS is LOW.
In PCM short mode the data transfer starts at the falling edge of WS, while it starts at the rising edge of WS in PCM long mode.
1: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS rising edge, and the right channel one CK cycle after the WS falling edge.
In MSB- or LSB-justified mode, the left channel is transfered when WS is LOW, and right channel when WS is HIGH.
In PCM short mode the data transfer starts at the rising edge of WS, while it starts at the falling edge of WS in PCM long mode.

Bit 12 **FIXCH**: fixed channel length in slave
0: the channel length in slave mode is different from 16 or 32 bits (CHLEN must be set)
1: the channel length in slave mode is supposed to be 16 or 32 bits (according to CHLEN)

Bit 11 **CKPOL**: serial audio clock polarity
0: the signals generated by the SPI/I2S (that is SDO and WS) are changed on the falling edge of CK and the signals received by the SPI/I2S (that is SDI and WS) are read of the rising edge of CK.
1: the signals generated by the SPI/I2S (that is SDO and WS) are changed on the rising edge of CK and the signals received by the SPI/I2S (that is SDI and WS) are read of the falling edge of CK.

Bit 10 **CHLEN**: channel length (number of bits per audio channel)
0: 16-bit wide
1: 32-bit wide

Bits 9:8 **DATLEN[1:0]**: data length to be transferred.
00: 16-bit data length
01: 24-bit data length
10: 32-bit data length
11: Not allowed
Note: Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), refer to Section 48.3: SPI implementation to check the supported data size.

Bit 7 **PCMSYNC**: PCM frame synchronization
0: short frame synchronization
1: long frame synchronization

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **I2SSTD[1:0]**: I²S standard selection
For more details on I²S standards, refer to *Section 48.9.5: Supported audio protocols*
00: I²S Philips standard.
01: MSB-justified standard (left justified)
10: LSB-justified standard (right justified)
11: PCM standard

Bits 3:1 **I2SCFG[2:0]**: I2S configuration mode
000: slave - transmit
001: slave - receive
010: master - transmit
011: master - receive
100: slave - Full Duplex
101: master - Full Duplex
others: not used

Bit 0 **I2SMOD**: I2S mode selection
0: SPI mode is selected
1: I2S/PCM mode is selected

48.11.15 SPI/I2S register map

Table 563. SPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SPI_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK	TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
	Reset value																0	0	0	0	0	0	0	0	0								0
0x004	SPI_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSIZE[15:0](1)															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	SPI_CFG1	BPASS	MBR[2:0]			Res.	Res.	Res.	DRDS	Res.	CRCEN	Res.	CRCSIZE[4:0](1)					TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0](1)				DSIZE[4:0](1)				
	Reset value	0	0	0	0				0		0		0	0	1	1	1	0	0					0	0	0	0	0	0	0	1	1	1
0x00C	SPI_CFG2	AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.	IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0		0	0	0						0	0	0	0	0	0	0	0
0x010	SPI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x014	SPI_SR	CTSIZE[15:0](1)																RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DPXP	TXP	RXP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0		0	0	0	0	0	0	0	0	1	0
0x018	SPI_IFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
	Reset value																					0		0	0	0	0	0	0	0			
0x01C	Reserved	Res.																															
0x020	SPI_TXDR	TXDR[31:16]																TXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 563. SPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024-0x02C	Reserved	Res.																															
0x030	SPI_RXDR	RXDR[31:16]																RXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034 - 0x03C	Reserved	Res.																															
0x040	SPI_CRCPOLY	CRCPOLY[31:16](2)																CRCPOLY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1
0x044	SPI_TXCRC	TXCRC[31:16](2)																TXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	SPI_RXCRC	RXCRC[31:16](2)																RXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	SPI_UDRDR	UDRDR[31:16](2)																UDRDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	SPI_I2SCFGR	Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]								Res.	Res.	WSINV	FIXCH	CKPOL	CHLEN	DATLEN[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]			I2SMOD
	Reset value							0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0		0	0	0	0	0	0

1. The configuration of this bitfield depends on the features of the SPI instance. For more details, refer to *Section 48.3: SPI implementation*.
2. The bits 31-16 are reserved for the peripheral instances with data size limited to 16 bits. There is no constraint when the 32-bit access is applied at these addresses. The bits 31-16, when reserved, are always read to zero while any write to them is ignored.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

49 FD controller area network (FDCAN)

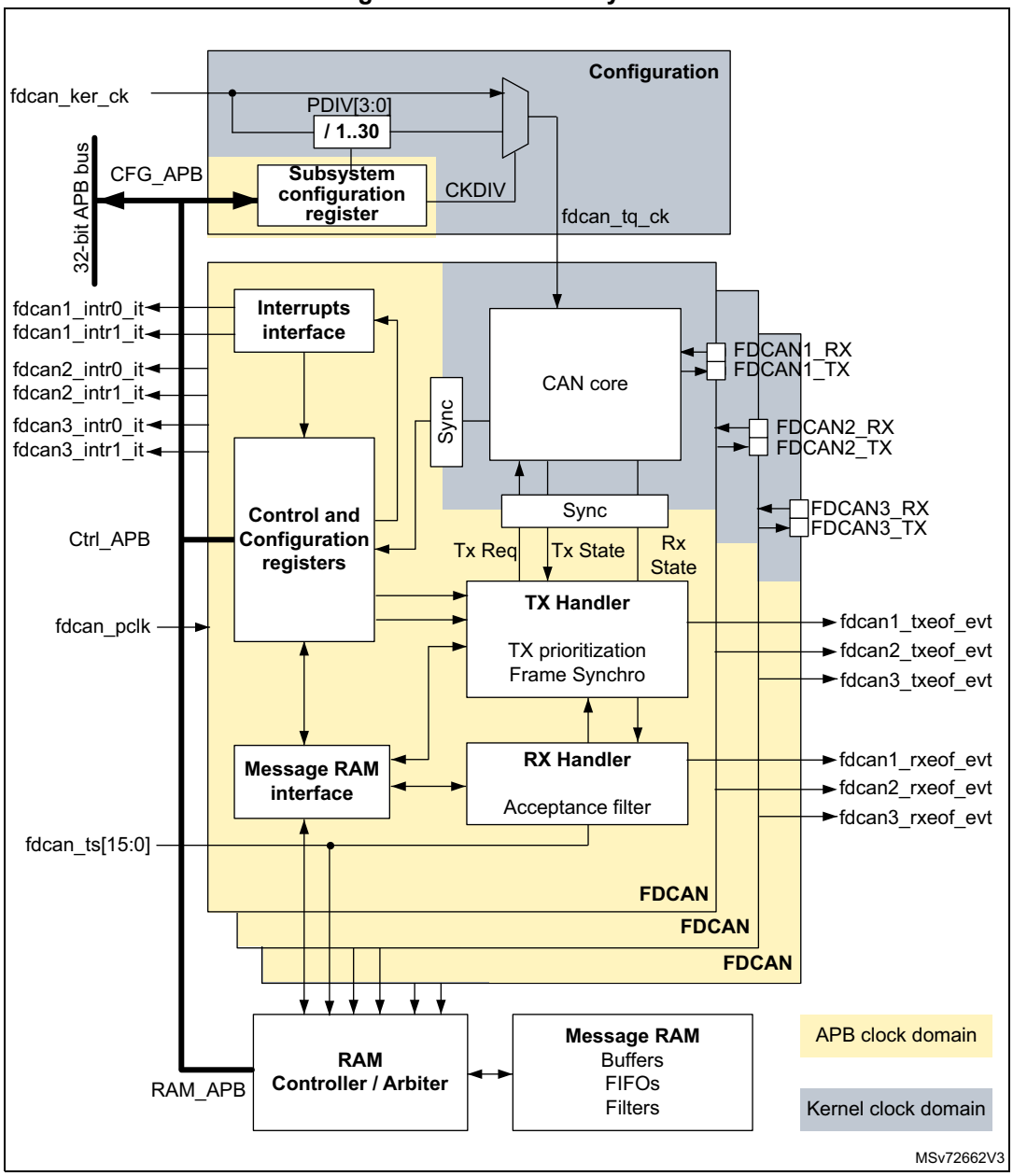
49.1 FDCAN introduction

The controller area network (CAN) subsystem (see *Figure 1000*) consists of three CAN modules, a shared message RAM, and a configuration block. Refer to the memory map for the base address of each of these parts.

The modules (FDCAN) are compliant with ISO 11898-1: 2015 (CAN protocol specification version 2.0 part A, B) and CAN FD protocol specification version 1.0.

A 0.8-Kbyte message RAM per FDCAN instance is used for filtering, transmitting event FIFOs, and receiving and transmitting FIFOs.

Figure 1000. CAN subsystem



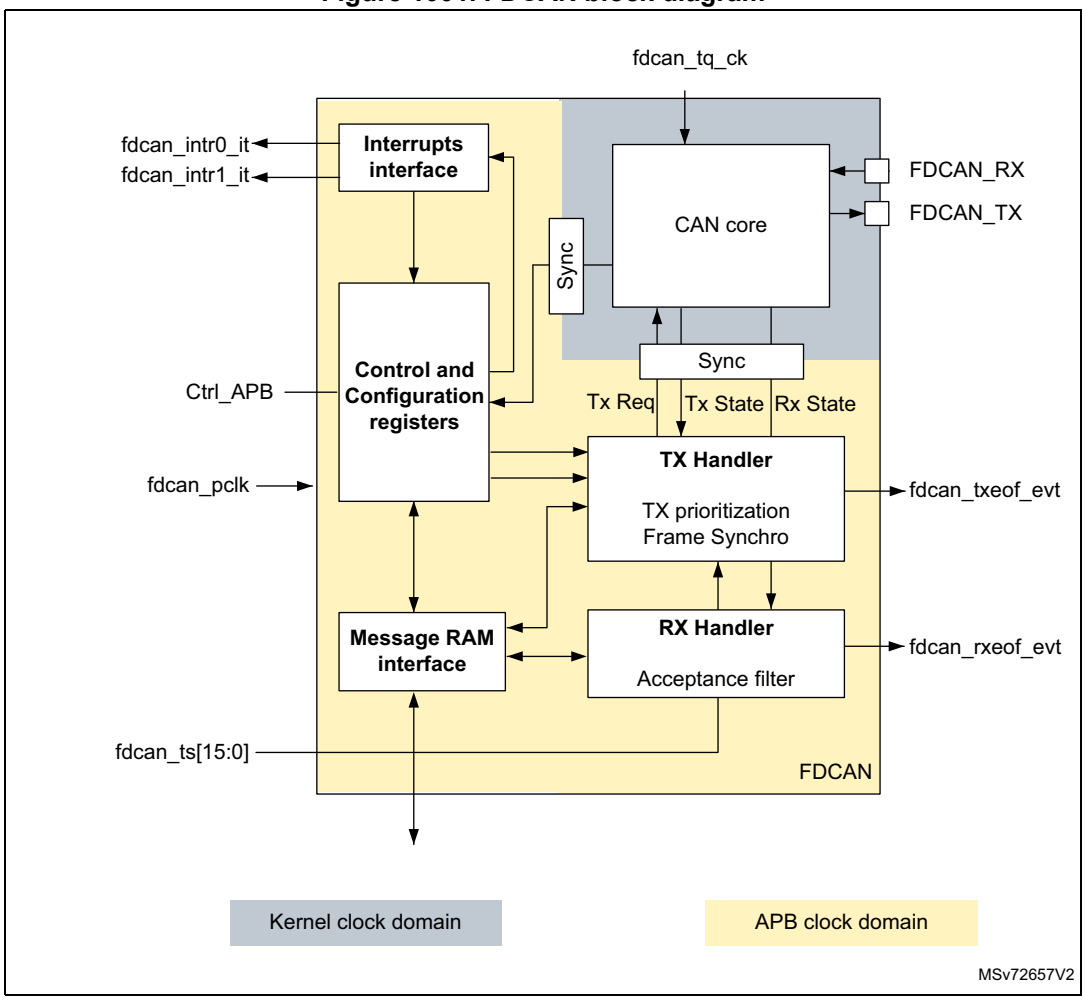
49.2 FDCAN main features

- Conform with CAN protocol version 2.0 part A, B, and ISO 11898-1: 2015
- CAN FD with maximum 64 data bytes supported
- CAN error logging
- AUTOSAR and J1939 support
- Improved acceptance filtering
- Two receive FIFOs of three payloads each (up to 64 bytes per payload)
- Separate signaling on reception of high priority messages
- Configurable transmit FIFO/queue of three payloads (up to 64 bytes per payload)
- Transmit event FIFO
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains: APB bus interface and CAN core kernel clock
- Power-down support

49.3 FDCAN functional description

49.3.1 FDCAN block diagram

Figure 1001. FDCAN block diagram



Dual interrupt lines

The FDCAN peripheral provides two interrupt lines, fdcan_intr0_it and fdcan_intr1_it.

By programming the EINT0 and EINT1 bits of the FDCAN_ILE register, the interrupt lines can be independently enabled or disabled.

CAN core

The CAN core contains the protocol controller and receive/transmit shift registers. It handles all ISO 11898-1: 2015 protocol functions and supports both 11-bit and 29-bit identifiers.

Sync

This block synchronizes signals from the APB clock domain to the CAN kernel clock domain and vice versa.

Tx handler

The Tx handler controls the message transfer from the message RAM to the CAN core. A maximum of three Tx buffers is available for transmission. The Tx buffer can be used as Tx FIFO or as a Tx queue. Tx event FIFO stores Tx timestamps together with the corresponding message ID. Transmit cancellation is also supported.

The Tx handler also generates an event on the fdcan_txeof_evt output when the end-of-frame transmission is issued. This event is internally connected to the TIM2 and TIM3 input capture.

Rx handler

The Rx handler controls the transfer of received messages from the CAN core to the external message RAM. The Rx handler supports two receive FIFOs, for storage of all messages that have passed acceptance filtering. An Rx timestamp is stored together with each message. Up to 28 filters can be defined for 11-bit IDs; up to eight filters for 29-bit IDs.

The Rx handler also generates an event on the fdcan_rxeof_evt output when the end-of-frame reception is received. This event is internally connected to the TIM2 and TIM3 input capture.

APB interface

The APB interface connects the FDCAN to the APB bus for configuration registers, controller configuration, and RAM access.

Message RAM interface

The message RAM interface connects the FDCAN access to an external 1-Kbyte message RAM through a RAM controller/arbiter.

49.3.2 FDCAN pins and internal signals

The CAN subsystem I/O signals and pins are detailed, respectively, in *Table 564*, *Table 565*, and *Figure 1000*.

Table 564. CAN subsystem I/O signals

Name	Type	Description
fdcan_ker_ck	Digital input	CAN subsystem kernel clock input
fdcan_pclk		CAN subsystem APB interface clock input
fdcan*n*_intr0_it	Digital output	FDCAN interrupt0
fdcan*n*_intr1_it		FDCAN interrupt1
fdcan_ts[15:0]	-	External timestamp vector
fdcan*n*_txeof_evt	Digital output	End-of-frame transmission event
fdcan*n*_rxeof_evt	Digital output	End-of-frame reception event
APB interface	Digital input/output	Single APP with multiple psel for configuration, control and RAM access

Table 565. CAN subsystem I/O pins

Name	Type	Description
FDCAN*n*_RX	Digital input	FDCAN receive pin
FDCAN*n*_TX	Digital output	FDCAN transmit pin

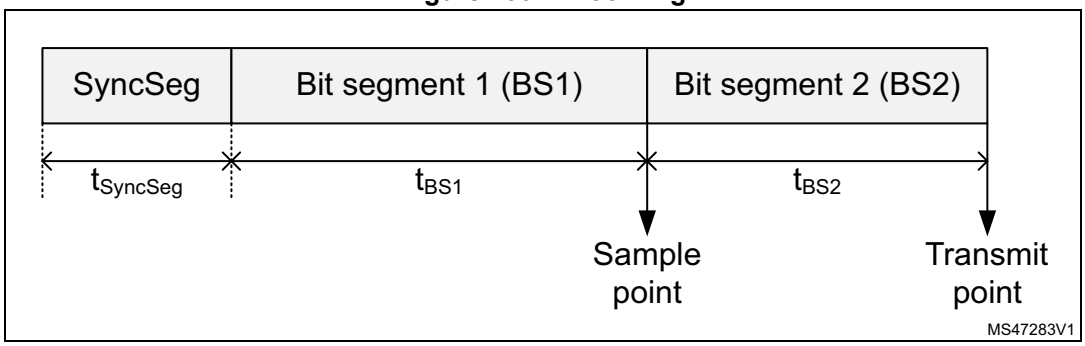
49.3.3 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

As shown in *Figure 1002*, this operation can be explained simply by splitting the bit time in three segments, as follows:

- Synchronization segment (SYNC_SEG): a bit change is expected to occur within this time segment, having a fixed length of one time quantum ($1 \times t_q$).
- Bit segment 1 (BS1): defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard. Its duration is programmable from 1 to 32 time quanta, but can be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of various nodes of the network.
- Bit segment 2 (BS2): defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard, its duration is programmable between 1 and 16 time quanta, but can also be automatically shortened to compensate for negative phase drifts.

Figure 1002. Bit timing



The baud rate is the inverse of the bit time (baud rate = 1 / bit time), which, in turn, is the sum of three components (see *Figure 1002*):

bit time = $t_{SyncSeg} + t_{BS1} + t_{BS2}$

Where:

– For the nominal bit time

$t_q = (NBRP[8:0] + 1) \times t_{fdcan_tq_clk}$

$t_{SyncSeg} = 1 \times t_q$

$t_{BS1} = t_q \times (NTSEG1[7:0] + 1)$

$t_{BS2} = t_q \times (NTSEG2[6:0] + 1)$

Where NBRP[8:0], NTSEG1[7:0], and NTSEG2[6:0] bitfields belong to the FDCAN_NBTP register.

– For the data bit time
  $t_q = (DBRP[4:0] + 1) \times t_{fdcan_tq_clk}$
  $t_{SyncSeg} = 1 \times t_q$
  $t_{BS1} = t_q \times (DTSEG1[4:0] + 1)$
  $t_{BS2} = t_q \times (DTSEG2[3:0] + 1)$
  Where DBRP[4:0], DTSEG1[4:0], and DTSEG2[3:0] belong to the FDCAN_DBTP register.

The (re)synchronization jump width (SJW) defines an upper bound for the amount of lengthening or shortening of the bit segments. It is programmable between one and four time quanta.

A valid edge is defined as the first transition in a bit time from recessive to dominant bus level, provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW, so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW, so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the bit timing register is only possible while the device is in Standby mode. The FDCAN_DBTP and FDCAN_NBTP registers (dedicated, respectively, to data and nominal bit timing) are only accessible when the CCE and INIT of the FDCA_CCCR register are set.

The FDCAN requires that the CAN time quanta clock is always below or equal to the APB clock ($f_{dcan_tq_ck} \leq f_{dcan_pclk}$).

Note: For a detailed description of the CAN bit timing and resynchronization mechanism, refer to the ISO 11898-1 standard.

49.3.4 Operating modes

Configuration

Access to peripheral version, hardware, and input clock divider configuration. When the clock divider is set to 0, the primary input clock is used as it is.

The reachable FDCAN bitrates are:

- Nominal bitrate: up to 1 Mbit/s
- Data bitrate: between nominal bitrate and 8 Mbit/s

Software initialization

Software initialization is started by setting the INIT bit of the FDCAN_CCCR register, by software, by a hardware reset, or by entering bus-off state. While the INIT bit is set, message transfers from and to the CAN bus are stopped, and the status of the CAN bus output FDCAN_TX is recessive (high). The EML (error management logic) counters are unchanged. Setting the INIT bit does not change any configuration register. Clearing INIT bit of FDCAN_CCCR finishes the software initialization. Afterwards the bit stream processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (bus-idle) before it can take part in bus activities and start the message transfer.

Access to the FDCAN configuration registers is only enabled when the INIT bit and the CCE bit of the FDCAN_CCCR register are both set.

The CCE bit of the FDCAN_CCCR register can only be set/cleared while the INIT bit of FDCAN_CCCR is set. The CCE bit is automatically cleared when the INIT bit is cleared.

The following registers are reset when the CCE bit of the FDCAN_CCCR register is set:

- FDCAN_HPMS: High priority message status
- FDCAN_RXF0S: Rx FIFO 0 status
- FDCAN_RXF1S: Rx FIFO 1 status
- FDCAN_TXFQS: Tx FIFO/queue status
- FDCAN_TXBRP: Tx buffer request pending
- FDCAN_TXBTO: Tx buffer transmission occurred
- FDCAN_TXBCF: Tx buffer cancellation finished
- FDCAN_TXEFS: Tx event FIFO status

The timeout counter value (TOC[15:0] bit of the FDCAN_TOCV register) is preset to the value configured by the TOP[15:0] of the FDCAN_TOCC register when the CCE bit of the FDCAN_CCCR is set.

In addition, the state machines of the Tx handler and Rx handler are held in idle state while the CCE bit is set.

The following registers can be written only when the CCE bit is cleared:

- FDCAN_TXBAR: Tx buffer add request
- FDCAN_TXBCR: Tx buffer cancellation request

The TEST and the MON bits of the FDCAN_CCCR register can be set only by software while the INIT and the CCE bits of the FDCAN_CCCR register are both set. Both bits can be reset at any time. The DAR bit of FDCAN_CCCR can only be set/cleared while the INIT and CCE bits are both set.

Normal operation

The FDCAN default operating mode after hardware reset is event-driven CAN communication. TT operation mode is not supported.

Once the FDCAN is initialized and the INIT bit of the FDCAN_CCCR register is cleared, the FDCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including message ID and DLC are stored into the Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted, the Tx FIFO or the Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not supported.

CAN FD operation

There are two variants in the FDCAN protocol:

- Long frame mode (LFM), where the data field of a CAN frame may be longer than eight bytes.
- Fast frame mode (FFM), where the control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate compared to the beginning and to the end of the frame.

The fast frame mode can be used in combination with the long frame mode.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers are decoded as FDF bit: FDF recessive signifies a CAN FD frame, while FDF dominant signifies a classic CAN frame.

In a CAN FD frame, the two bits following FDF (res and BRS) decide whether the bit rate inside this CAN FD frame is switched. A CAN FD bit rate switch is signified by res dominant and BRS recessive. The coding of res recessive is reserved for future expansion of the protocol. In case the FDCAN receives a frame with FDF recessive and res recessive, it signals a protocol exception event by setting the PXE bit of the FDCAN_PSR register. When protocol exception handling is enabled (PXHD = 0 in FDCAN_CCCR), this causes the operation state to change from receiver (ACT[1:0] = 10 in FDCAN_PSR) to integrating (ACT[1:0] = 00 in FDCAN_PSR) at the next sample point. If protocol exception handling is disabled (PXHD = 1 in FDCAN_CCCR), the FDCAN treats a recessive res bit as a form error and responds with an error frame.

CAN FD operation is enabled by programming the FDOE bit of the FDCAN_CCCR register. In case FDOE = 1, transmission and reception of CAN FD frames are enabled. Transmission and reception of classic CAN frames are always possible. Whether a CAN FD frame or a classic CAN frame is transmitted can be configured via the FDF bit in the respective Tx buffer element. With FDOE = 0, received frames are interpreted as classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if the FDF bit of a Tx buffer element is set. The FDOE and BRSE bits of the FDCAN_CCCR register can only be changed while the INIT and CCE bits are both set.

With FDOE = 0, the setting of the FDF and BRS bits is ignored, and frames are transmitted in classic CAN format. With FDOE = 1 and BRSE = 0, only the FDF bit of a Tx buffer element is evaluated. With FDOE = 1 and BRSE = 1, transmission of CAN FD frames with bit rate switching is enabled. All Tx buffer elements with FDF and BRS bits set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is recommended only under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case, disable the CAN FD bit rate switching option for transmissions.
- During system startup, all nodes transmit classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN partial networking have to be transmitted in classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non-CAN FD nodes are held in silent mode until programming is complete. Then all nodes switch back to classic CAN communication.

In the FDCAN format, the coding of the DLC differs from that of the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15 (that in standard CAN all code a data field of 8 bytes) are coded according to *Table 566*.

Table 566. DLC coding in FDCAN

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD fast frames, the bit timing is switched inside the frame, after the BRS (bit rate switch) bit, if this bit is recessive. Before the BRS bit, in the FDCAN arbitration phase, the standard CAN bit timing is used as defined by the FDCAN_DBTP register. In the following FDCAN data phase, the fast CAN bit timing is used as defined by the FDCAN_DBTP register. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the FDCAN kernel clock frequency. For example, with an FDCAN kernel clock frequency of 20 MHz and the shortest configurable bit time of four time quanta (t_q), the bit rate in the data phase is 5 Mbit/s.

In both data frame formats (CAN FD long frames and CAN FD fast frames), the value of bit ESI (error status indicator) is determined by the transmitter error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames, the ESI bit is always transmitted dominant, independent of the transmitter error state. The data length code of CAN FD remote frames is transmitted as 0.

In case an FDCAN Tx buffer is configured for FDCAN transmission with DLC > 8, the first eight bytes are transmitted as configured in the Tx buffer while the remaining part of the data field is padded with 0xCC. When the FDCAN receives a FDCAN frame with DLC > 8, the first eight bytes of that frame are stored into the matching Rx FIFO. The remaining bytes are discarded.

Transceiver delay compensation

During the data phase of an FDCAN transmission, only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin FDCAN_TX, the protocol controller receives the transmitted data from its local CAN transceiver via pin FDCAN_RX. The received data is delayed by the CAN transceiver loop delay. If this delay is greater than TSEG1 (time segment before sample point), and a bit error is detected. Without transceiver delay compensation, the bit rate in the data phase of an FDCAN frame is limited by the transceiver loop delay.

The FDCAN implements a delay compensation mechanism to compensate the CAN transceiver loop delay, thereby enabling transmission with higher bit rates during the FDCAN data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the secondary sample point (SSP). If a bit error is detected, the transmitter reacts on this bit error at the next following regular sample point. During the arbitration phase, the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay. This is enabled by setting the TDC bit of the FDCAN_DBTP register, and described in detail in the ISO11898-1 specification.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the FDCAN transmit output pin FDCAN_TX through the transceiver to the receive input pin FDCAN_RX plus the transmitter delay compensation offset as configured by TDCO[6:0] of FDCAN_TDCR. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example, half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mt_q (minimum time quantum, one period of fdcan_tq_ck clock).

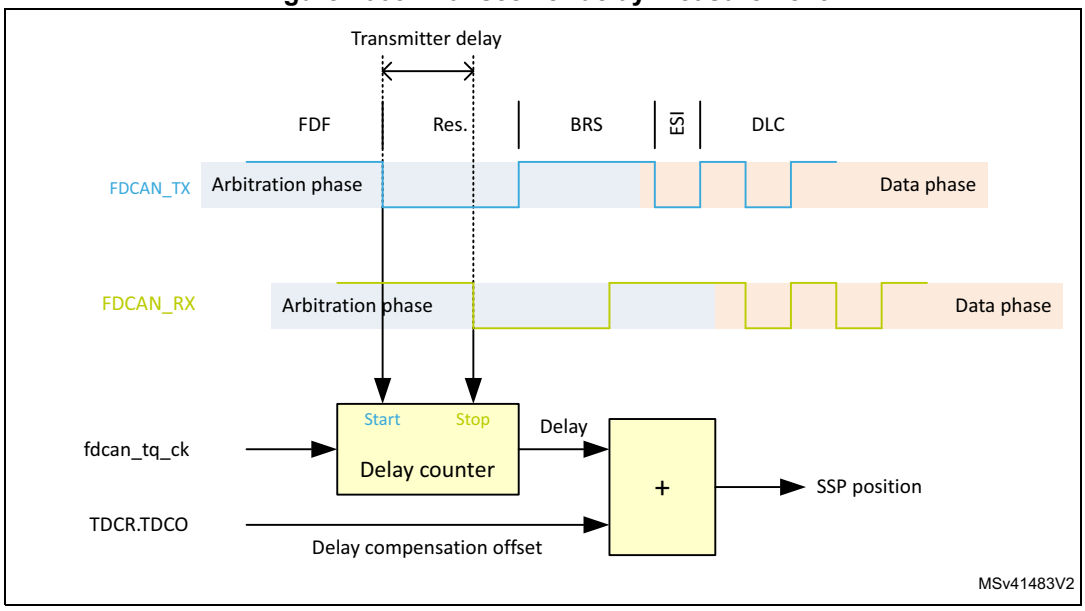
The TDCV[6:0] bitfield of the FDCAN_PSR register shows the actual transmitter delay compensation value. TDCV[6:0] is cleared when the INIT is set in the FDCAN_CCCR. It is updated at each transmission of an FD frame while the TDC bit of the FDCAN_DBTP register is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the FDCAN:

- The sum of the measured delay from FDCAN_Tx to FDCAN_Rx and the configured transmitter delay compensation offset TDCO[6:0] has to be lower than 6-bit times in the data phase.
- The sum of the measured delay from FDCAN_TX to FDCAN_RX and the configured transmitter delay compensation offset TDCO[6:0] has to be lower than or equal to 127 × mt_q. If the sum exceeds this value, the maximum value (127 × mt_q) is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, which stops checking received bits at the SSPs.

If transmitter delay compensation is enabled by setting the TDC bit of the FDCAN_DBTP; the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input pin FDCAN_TX of the transmitter. The resolution of this measurement is one mt_q.

Figure 1003. Transceiver delay measurement



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit (resulting in a too early SSP position), the use of a transmitter delay compensation filter window can be enabled by programming the TDCF[6:0] bitfield of the FDCAN_TDCR register. This defines a minimum value for the SSP position. Dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least TDCF[6:0] and FDCAN_RX is low.

Restricted operation mode

In restricted operation mode, the node is able to receive data and remote frames, and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits. Instead, it waits for the occurrence of a bus-idle condition to resynchronize itself to the CAN communication. The error counters (REC[6:0] and TEC[7:0] in FDCAN_ECR) are frozen while the error logging (CEL[7:0]) is active. The software can set the FDCAN into restricted operation mode by setting the ASM bit of FDCAN_CCCR. The bit can only be set by software when both CCE and INIT bits are set in FDCAN_CCCR. The bit can be cleared by software at any time.

Restricted operation mode is automatically entered when the Tx handler is not able to read data from the message RAM in time. To leave restricted operation mode, the software has to clear the ASM bit of FDCAN_CCCR.

The restricted operation mode can be used in applications that adapt themselves to different CAN bit rates. In this case, the application tests different bit rates and leaves the restricted operation mode after it has received a valid frame.

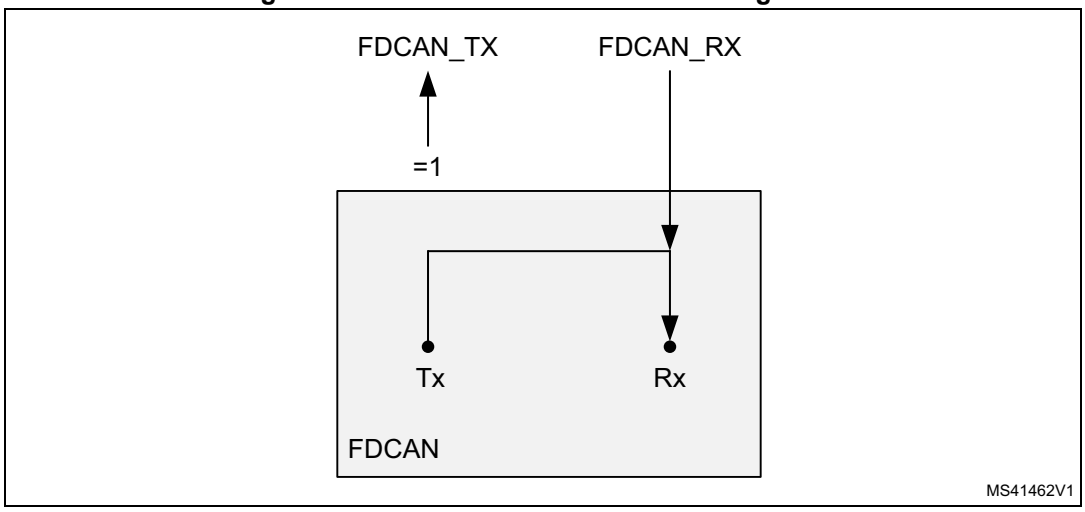
Note: The restricted operation mode must not be combined with the loop-back mode (internal or external).

Bus monitoring mode

The FDCAN is set in bus monitoring mode by setting the MON bit of the FDCAN_CCCR register. In bus monitoring mode (for more details refer to ISO11898-1, 10.12 bus monitoring), the FDCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the FDCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the FDCAN can monitor it, even if the CAN bus remains in recessive state. In bus monitoring mode, the FDCAN_TXBRP register is held in reset state.

The bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. *Figure 1004* shows the connection of FDCAN_TX and FDCAN_RX signals to the FDCAN in bus monitoring mode.

Figure 1004. Pin control in bus monitoring mode



Disabled automatic retransmission mode (DAR)

According to the CAN specification (see ISO11898-1, 6.3.3 Recovery Management), the FDCAN provides means for automatic retransmission of frames that have lost arbitration or have been disturbed by errors during transmission. By default, automatic retransmission is enabled. The DAR mode can be disabled through the DAR bit of the FDCAN_CCCR register.

Frame transmission in DAR mode

In DAR mode, all transmissions are automatically canceled after they have been started on the CAN bus. A Tx buffer Tx request pending bit (TRPx in FDCAN_TXBRP) is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, when it has been aborted due to lost arbitration, or when an error has occurred during frame transmission.

- Successful transmission
  - The corresponding Tx buffer transmission occurred bit TOx is set in FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is cleared in the FDCAN_TXBCF register.
- Successful transmission in spite of cancellation
  - The corresponding Tx buffer transmission occurred bit TOx is set in the FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is set in the FDCAN_TXBCF register.
- Arbitration loss or frame transmission disturbed
  - The corresponding Tx buffer transmission occurred bit TOx is cleared in the FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is set in the FDCAN_TXBCF register.

In case of a successful frame transmission, and if the storage of Tx events is enabled, a Tx event FIFO element is written with event type ET = 10 (transmission in spite of cancellation).

Power-down (Sleep mode)

Power-down entry

The FDCAN can be set into power-down mode controlled by setting the CSR bit of the FDCAN_CCCR register. As long as the clock stop request is active, CSR is read as 1.

When all pending transmission requests have completed, the FDCAN waits until the bus-idle state is detected. The FDCAN then sets the INIT bit of the FDCAN_CCCR register to prevent any further CAN transfers. Now, the FDCAN acknowledges that it is ready for power-down by setting the CSA bit of the FDCAN_CCCR register. In this state, before the clocks are switched off, further register accesses can be made. A write access to the INIT bit has no effect. Now, the module clock inputs can be switched off.

Power-down exit

To leave power-down mode, the application has to turn on the module clocks before clearing the CSR bit. The FDCAN acknowledges this by clearing the CSA bit. Afterwards, the application can restart CAN communication by clearing the INIT bit.

Test modes

To enable write access to *FDCAN test register (FDCAN_TEST)*, the TEST bit of the FDCAN_CCCR register must be set, thus enabling the configuration of test modes and functions.

Four output functions are available for the CAN transmit pin FDCAN_TX by programming the TX[1:0] bitfield of the FDCAN_TEST register. In addition to its default function (the serial data output), it can drive the CAN sample point signal to monitor the FDCAN bit timing as well as drive constant dominant or recessive values. The actual value at pin FDCAN_RX can be read from the RX bit of FDCAN_TEST. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN kernel clock and APB clock domain, there can be a delay of several APB clock periods between writing to TX[1:0] until the new configuration is visible at FDCAN_TX output pin. This applies also when reading FDCAN_RX input pin via RX.

Note: Test modes must be used for production tests or self-test only. The software control for FDCAN_TX pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

External loop-back mode

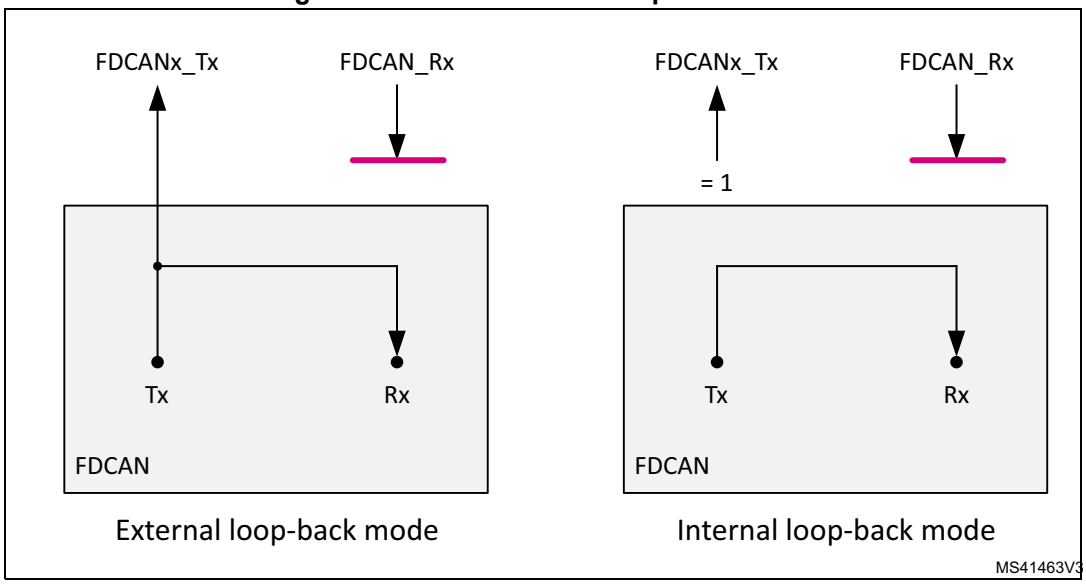
The FDCAN can be set in external loop-back mode by setting the LBCK bit of the FDCAN_TEST register. In loop-back mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 1005* shows the connection of transmit and receive signals FDCAN_TX and FDCAN_RX to the FDCAN in external loop-back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the FDCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loop-back mode. In this mode, the FDCAN performs an internal feedback from its transmit output to its receive input. The actual value of the FDCAN_RX input pin is disregarded by the FDCAN. The transmitted messages can be monitored at the FDCAN_TX transmit pin.

Internal loop-back mode

Internal loop-back mode is entered by setting both the LBCK bit of FDCAN_TEST and the MON bit of FDCAN_CCR. This mode can be used for a “hot self-test”, meaning the FDCAN can be tested without affecting a running CAN system connected to the FDCAN_TX and FDCAN_RX pins. In this mode, FDCAN_RX pin is disconnected from the FDCAN and FDCAN_TX pin is held recessive. *Figure 1005* shows the connection of FDCAN_TX and FDCAN_RX pins to the FDCAN in case of internal loop-back mode.

Figure 1005. Pin control in loop-back mode



Timestamp generation

For timestamp generation, the FDCAN supplies a 16-bit wrap-around counter. A prescaler (TCP[3:0] of FDCAN_TSCC) can be configured to clock the counter in multiples of CAN bit times (1 to 16). The counter is readable via the TCV[15:0] bitfield of the FDCAN_TSCV register. A write access to TSCV15:0] resets the counter to 0. When the timestamp counter wraps around, the interrupt flag (TSW bit of FDCAN_ISR) is set.

On start of frame reception/transmission, the counter value is captured and stored into the timestamp section of an Rx FIFO (RXTS[15:0]) or Tx event FIFO (TXTS[15:0]) element.

By programming TSS[1:0] of FDCAN_TSCC, a 16-bit timestamp can be used.

Debug mode behavior

In debug mode, the set/reset on read feature is automatically disabled during the debugger register access, and enabled during normal MCU operation

Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the FDCAN supplies a 16-bit timeout counter. It operates as a down-counter and uses the same prescaler controlled by TCP[3:0] of FDCAN_TSCC as the timestamp counter. The timeout counter is configured via the FDCAN_TOCC register. The actual counter value can be read from the TOC[15:0] bitfield of FDCAN_TOCV. The timeout counter can only be started while the INIT bit of FDCAN_CCCR is cleared. It is stopped when INIT is set, for example, when the FDCAN enters bus-off state.

The operation mode is selected by TOS[1:0] of FDCAN_TOCC. When operating in continuous mode, the counter starts when INIT is cleared. A write to FDCAN_TOCV presets the counter to the value configured by TOP[15:0] in FDCAN_TOCC and continues down-counting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP[15:0]. Down-counting is started when the first FIFO element is stored. Writing to FDCAN_TOCV has no effect.

When the counter reaches 0, the TOO interrupt flag is set in the FDCAN_IR register. In continuous mode, the counter is immediately restarted at TOP[15:0].

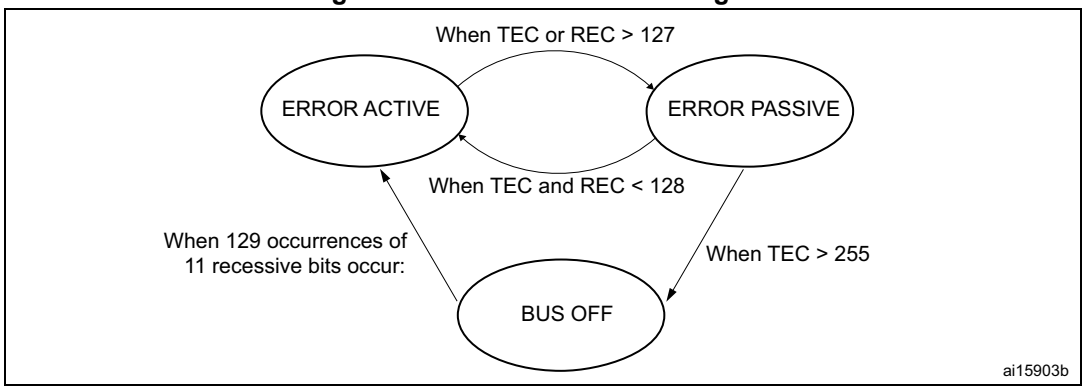
Note: The clock signal for the timeout counter is derived from the CAN core sample point signal. Therefore, the point in time where the timeout counter is decremented may vary due to the synchronization/resynchronization mechanism of the CAN core. If the baud rate switch feature in FDCAN is used, the timeout counter is clocked differently in the arbitration and data fields.

49.3.5 Error management

As described in the CAN protocol, the error management is handled entirely by hardware using the transmit error counter (the TEC[7:0] bitfield of the *FDCAN error counter register (FDCAN_ECR)*) and the receive error counter (the REC[6:0] bitfield of the *FDCAN error counter register (FDCAN_ECR)*). These values are incremented or decremented according to the error condition. For detailed information on TEC[7:0] and REC[6:0] management, refer to the CAN standard. Both values can be read by software to determine the stability of the network.

The bus-off state is reached when TEC[7:0] is greater than 255. This state is also indicated by the BO flag of the *FDCAN protocol status register (FDCAN_PSR)*. In bus-off state, the CAN is no longer able to transmit and receive messages. It has to wait at least for the duration of the recovery sequence specified in the CAN standard (129 occurrences of 11 consecutive recessive bits monitored on FDCAN_RX input).

Figure 1006. CAN error state diagram

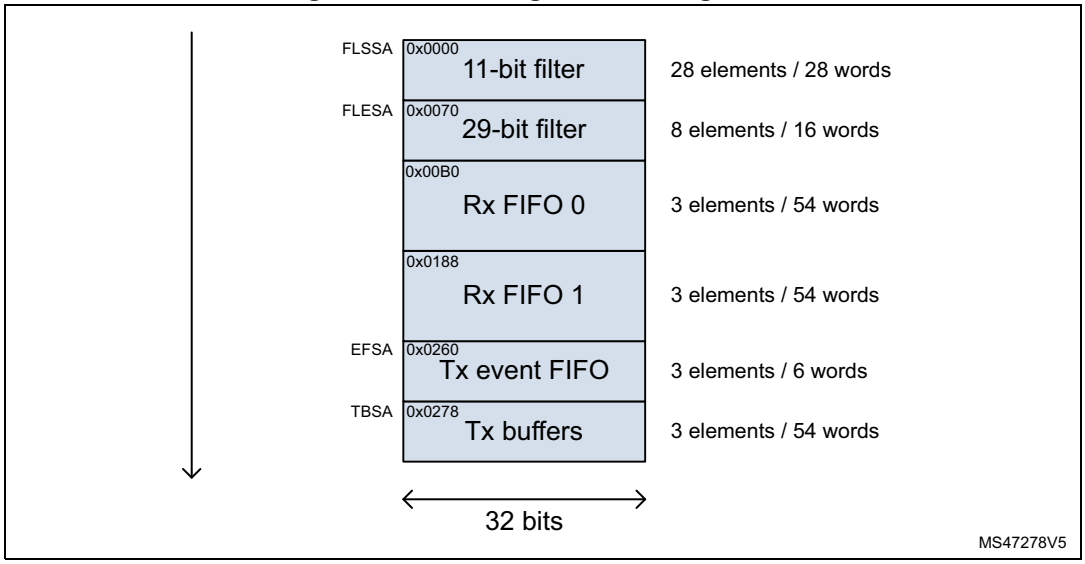


Note: In initialization mode, the CAN does not monitor the FDCAN_RX *signal, and therefore cannot complete the recovery sequence. To recover from an error state, the CAN must operate in normal mode.*

49.3.6 Message RAM

The message RAM is 32-bit wide, and the FDCAN module is configured to allocate up to 212 words in it. It is not necessary to configure each of the sections shown in *Figure 1007*.

Figure 1007. Message RAM configuration



When the FDCAN addresses the message RAM, it addresses 32-bit words (aligned), not a single byte. The RAM addresses are 32-bit words, that is, only bits 15 to 2 are evaluated, the two least significant bits are ignored.

In case of multiple instances, the RAM start address for the FDCANn is computed by end address + 4 of FDCANn - 1, and the FDCANn end address is computed by FDCANn start address + 0x0350 - 4.

As an example, for two instances:

- FDCAN1:
  - Start address 0x0000
  - End address 0x034C (as in *Figure 1007*)
- FDCAN2:
  - Start address = 0x034C (FDCAN1 end address) + 4 = 0x0350
  - End address = 0x0350 (FDCAN2 start address) + 0x0350 - 4 = 0x069C.

Rx handling

The Rx handler controls the acceptance filtering, the transfer of received messages to one of the two Rx FIFOs, as well as the Rx FIFO put and get indices.

Acceptance filter

The FDCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and another for extended identifiers. These filters can be assigned to Rx FIFO 0 or Rx FIFO 1. For acceptance filtering, each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element, and the following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - Range filter (from - to)
  - Filter for one or two dedicated IDs
  - Classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering.
- Each filter element can be enabled/disabled individually.
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global filter configuration (RXGFC)
- Extended ID AND mask (XIDAM)

Depending on the configuration of the filter element (SFEC[2:0]/EFEC[2:0]), a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Reject received frame
- Set the high priority message interrupt flag HPM in FDCAN_IR
- Set the high priority message interrupt flag HPM in FDCAN_IR, and store the received frame in FIFO 0 or FIFO 1.

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx FIFO has been found, the message handler starts writing the received message data in 32-bit portions to the matching Rx FIFO. If the CAN protocol controller has detected an error condition (for example, CRC error), this message is discarded with the following impact:

- **Rx FIFO**

  The put index of the matching Rx FIFO is not updated, but the related Rx FIFO element is partly overwritten with the received data. For error type, see LEC[2:0] and DLEC[2:0] bitfields of the FDCAN_PSR register. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in *Rx FIFO overwrite mode* have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, the unmodified received identifier is stored independently from the used filters. The result of the acceptance filter process strongly depends on the sequence of configured filter elements.

Range filter

The filter matches for all received frames with message IDs in the range defined by SF1ID/SF2ID and EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

- EFT[1:0] = 00: the message ID of received frames is AND-ed with the extended ID AND mask (XIDAM) before the range filter is applied.
- EFT[1:0] = 11: the extended ID AND mask (XIDAM) is not used for range filtering.

Filter for dedicated IDs

A filter element can be configured to filter for one or two specific message IDs. To filter for one specific message ID, the filter element has to be configured with SF1ID = SF2ID and EF1ID = EF2ID.

Classic bit mask filter

The classic bit mask filtering is intended to filter groups of message IDs by masking single bits of a received message ID. With classic bit mask filtering SF1ID/EF1ID is used as message ID filter, while SF2ID/EF2ID is used as filter mask.

0 bit at the filter mask masks out the corresponding bit position of the configured ID filter. For example, the value of the received message ID at that bit position is not relevant for acceptance filtering. Only the bits of the received message ID where the corresponding mask bits are 1 are relevant for acceptance filtering.

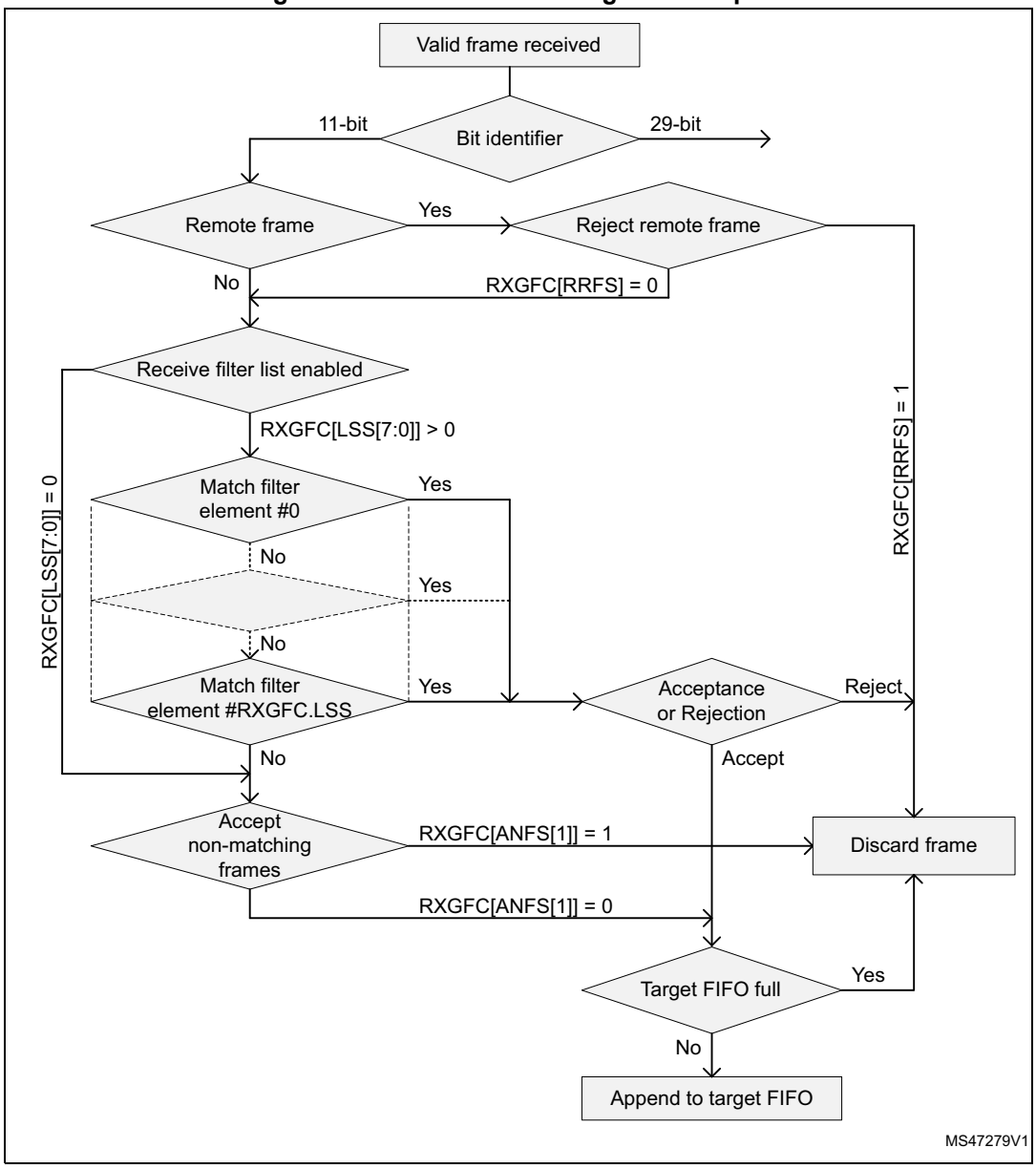
In case all mask bits are 1, a match occurs only when the received message ID and the message ID filter are identical. If all mask bits are 0, all message IDs match.

Standard message ID filtering

Figure 1008 shows the flow for standard message ID (11-bit identifier) filtering. The standard message ID filter element is described in *Section 49.3.11*.

The standard message filtering is controlled by the FDCAN_RXGFC register. The standard message ID, the remote transmission request bit (RTR), and the identifier extension bit (IDE) of the received frames are compared against the list of configured filter elements.

Figure 1008. Standard message ID filter path

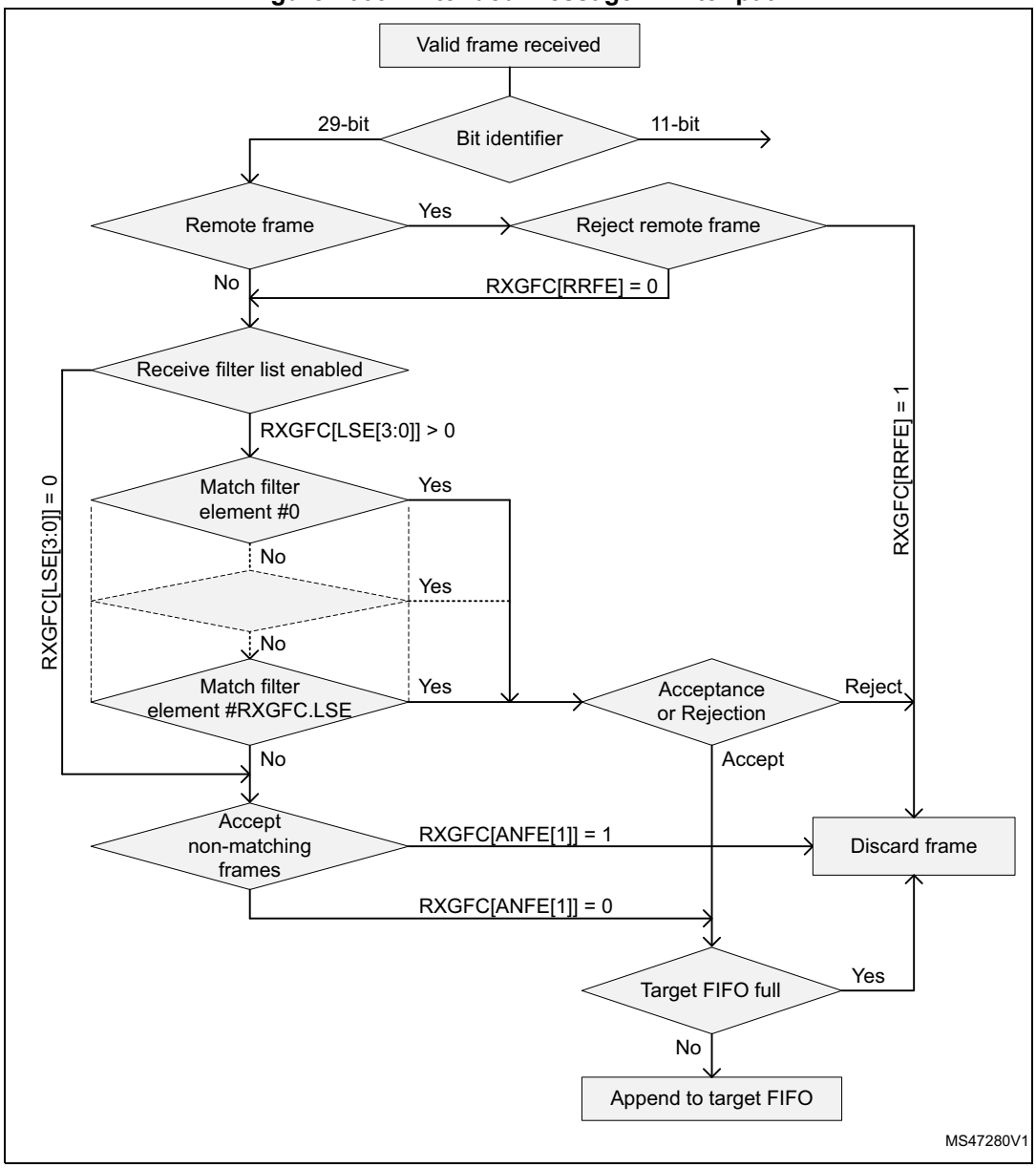


Extended message ID filtering

Figure 1009 shows the flow for extended message ID (29-bit identifier) filtering. The extended message ID filter element is described in *Section 49.3.12*.

The extended message filtering is controlled by the FDCAN_RXGFC register. The extended message ID, the remote transmission request bit (RTR), and the identifier extension bit (IDE) of the received frames are compared against the list of configured filter elements.

Figure 1009. Extended message ID filter path



The extended ID AND mask (XIDAM) is AND-ed with the received identifier before the filter list is executed.

Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can hold up to three elements each.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see *Acceptance filter*. The Rx FIFO element is described in *Section 49.3.8*.

When an Rx FIFO full condition is signaled by RFnF in FDCAN_IR (where n is the FIFO number), no further messages are written to the corresponding Rx FIFO until at least one message has been read out, and the Rx FIFO get index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded, and the interrupt flag RFnL is set in the FDCAN_IR register.

When reading from an Rx FIFO, the Rx FIFO get index (FnGI of FDCAN_RXFnS) + FIFO element size has to be added to the corresponding Rx FIFO start address (FnSA).

Rx FIFO blocking mode

The Rx FIFO blocking mode is configured by clearing the FnOM bit in the FDCAN_RXGFC register. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (FnPI = FnGI in FDCAN_RXFnS), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO get index has been incremented. An Rx FIFO full condition is signaled by FnF = 1 in FDCAN_RXFnS. In addition, the RFnF interrupt flag is set in FDCAN_IR.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded, and the message lost condition is signaled by setting RFnL bit in FDCAN_RXFnS. In addition, the RFnL interrupt flag is set in FDCAN_IR.

Rx FIFO overwrite mode

The Rx FIFO overwrite mode is configured by setting the FnOM bit of the FDCAN_RXGFC register.

When an Rx FIFO full condition (FnPI = FnGI of FDCAN_RXFnS) is signaled by FnF = 1 in FDCAN_RXFnS, the next message accepted for the FIFO overwrites the oldest FIFO message. Put and get indices are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading from the Rx FIFO elements must start at least at get index + 1. This is because it may happen that a received message is written to the message RAM (put index) while the CPU is reading from the message RAM (get index). In this case, inconsistent data can be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO.

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO acknowledge index (FnA of FDCAN_RXFnA). This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (FnF = 0 in FDCAN_RXFnS).

Tx handling

The Tx handler handles transmission requests for the Tx FIFO and the Tx queue. It controls the transfer of transmit messages to the CAN core, the put and get indices, and the Tx event FIFO. Up to three Tx buffers can be set up for message transmission. The CAN message data field is configured to 64 bytes. the Tx FIFO allocates eighteen 32-bit words for storage of a Tx element.

Table 567. Possible configurations for frame transmission

CCCR		Tx buffer element		Frame transmission
BRSE	FDOE	FDF	BRS	
Ignored	0	Ignored	Ignored	Classic CAN
0	1	0	Ignored	Classic CAN
0	1	1	Ignored	FD without bit rate switching
1	1	0	Ignored	Classic CAN
1	1	1	0	FD without bit rate switching
1	1	1	1	FD with bit rate switching

Note: AUTOSAR requires at least three Tx queue buffers and support of transmit cancellation.

The Tx handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the Tx buffer request pending register (FDCAN_TXBRP) is updated, or when a transmission has been started.

Transmit pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are permanently specified to specific values and cannot easily be changed. These message identifiers can have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority must be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

As an example, if CAN ECU-1 has the feature enabled and is requested by its application software to transmit four messages, it waits, after the first successful message transmission, for two CAN bit times of bus-idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, these messages are started in the idle time, and they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The feature is controlled by the TXP bit of the CCCR register. If the bit is set, the FDCAN, each time it has successfully transmitted a message, pauses for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. By default, this feature is disabled (TXP = 0 in FDCAN_CCCR).

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

Tx FIFO

Tx FIFO operation is configured by clearing the TFQM bit of the FDCAN_TXBC register. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the get index (TFGI[1:0] bitfield of FDCAN_TXFQS). After each transmission, the get index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same message ID from different Tx buffers in the order that these messages have been written to the Tx FIFO. The FDCAN calculates the Tx FIFO free level (TFFL[2:0] bitfield of FDCAN_TXFQS) as the difference between the get and put index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx buffer referenced by the put index (TFQPI[1:0] bitfield of FDCAN_TXFQS). An add request increments the put index to the next free Tx FIFO element. When the put index reaches the get index, Tx FIFO full (TFQF = 1 in FDCAN_TXFQS) is signaled. In this case, no further messages must be written to the Tx FIFO until the next message has been transmitted and the get index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by setting the FDCAN_TXBAR bit related to the Tx buffer referenced by the Tx FIFO put index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx buffers starting with the put index. The transmissions are then requested via the FDCA_TXBAR register. The put index is then cyclically incremented by n. The number of requested Tx buffers must not exceed the number of free Tx buffers as indicated by the Tx FIFO free level.

When a transmission request for the Tx buffer referenced by the get index is canceled, the get index is incremented to the next Tx buffer with a transmission request is pending and the Tx FIFO free level is recalculated. When transmission cancellation is applied to any other Tx buffer, the get index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates eighteen 32-bit words in the message RAM. The Therefore, the start address of the next available (free) Tx FIFO buffer, is calculated by adding 18 times the put index TFQPI[1:0] (0 … 2) to the Tx buffer start address TBSA.

Tx queue

Tx queue operation is configured by setting the TFQM of the FDCAN_TXBC register. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority).

In case of mixing of standard and extended message IDs, the standard message IDs are compared to bits [28:18] of extended message IDs.

In case multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx buffer referenced by the put index (TFQPI[1:0] in FDCAN_TXFQS). An add request cyclically increments the put index to the next free Tx buffer. In case the Tx queue is full (TFQF = 1 in FDCAN_TXFQS), the put index is not valid and no further message must be written to the Tx queue until at least one of the requested messages has been sent out or a pending transmission request has been canceled.

The application can use the FDCAN_TXBRP register instead of the put index and can place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocates eighteen 32-bit words in the message RAM. The start address of Therefore, the next available (free) Tx queue buffer is calculated by adding 18 times the Tx queue put index TFQPI[1:0] (0 ... 2) to the Tx buffer start address TBSA.

Transmit cancellation

The FDCAN supports transmit cancellation. To cancel a requested transmission from a Tx queue buffer, the host has to write 1 to the corresponding bit position (= number of Tx buffer) of the FDCAN_TXBCR register. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of the FDCAN_TXBCF register.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding FDCAN _TXBRP bit remains set as long as the transmission is in progress. If the transmission is successful, the corresponding FDCAN_TXBTO and FDCAN_TXBCF bits are set. If the transmission is not successful, it is not repeated and only the corresponding FDCAN_TXBCF bit is set.

Note: In case a pending transmission is canceled immediately before it has been started, there is a short time window where no transmission is started even if another message is pending in the node. This can enable another node to transmit a message that can have a priority lower than that of the second message in the node.

Tx event handling

To support Tx event handling the FDCAN has implemented a Tx event FIFO. After the FDCAN has transmitted a message on the CAN bus, message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the message marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO is configured to three elements. The Tx event FIFO element is described in *Tx FIFO*.

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling that is, a Tx buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by the TEFF bit of the FDCAN_IR, no further elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO get index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and the TEFL interrupt flag is set in the FDCAN_IR register.

When reading from the Tx event FIFO, the Tx event FIFO get index (EFGI[1:0] of FDCAN_TXEFS) has to be added twice to the Tx event FIFO start address EFSA.

49.3.7 FIFO acknowledge handling

The get indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO acknowledge index (see *Section 49.4.23* and *Section 49.4.25*). Writing to the FIFO acknowledge index sets the FIFO get index to the FIFO acknowledge index plus one and thereby updates the FIFO fill level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the get index), this get index value is written to the FIFO acknowledge index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO acknowledge index only once at the end of that read sequence (value = index of the last element read), to update the FIFO get index.

Because the CPU has free access to the FDCAN message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (get index not considered). This might be useful when reading a high priority message from one of the two Rx FIFOs. In this case, the FIFO acknowledge index must not be written because this would set the get index to a wrong position and alter the FIFO fill level. In this case, some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO acknowledge index. The FDCAN does not check for erroneous values.

49.3.8 FDCAN Rx FIFO element

Two Rx FIFOs are configured in the message RAM. Each Rx FIFO section can be configured to store up to three received messages. The structure of an Rx FIFO element is described in *Table 568*. The description is provided in *Table 569*.

Table 568. Rx FIFO element

Bit	31			24	23							16	15	8	7	0
R0	ESI	XTD	RTR	ID[28:0]												
R1	ANMF	FIDX[6:0]			Res.			FDF	BRS	DLC[3:0]			RXTS[15:0]			
R2	DB3[7:0]				DB2[7:0]								DB1[7:0]		D[7:0]	
R3	DB7[7:0]				DB6[7:0]								DB5[7:0]		DB4[7:0]	
⋮	⋮				⋮								⋮			
Rn	DBm[7:0]				DBm-1[7:0]								DBm-2[7:0]		DBm-3[7:0]	

The element size configured for storage of CAN FD messages is set to 64-byte data field.

Table 569. Rx FIFO element description

Field	Description
R0 Bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
R0 Bit 30 XTD	Extended identifier Signals to the host whether the received frame has a standard or extended identifier. – 0: 11-bit standard identifier – 1: 29-bit extended identifier

Table 569. Rx FIFO element description (continued)

Field	Description
R0 Bit 29 RTR	Remote transmission request Signals to the host whether the received frame is a data frame or a remote frame. – 0: Received frame is a data frame – 1: Received frame is a remote frame
R0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 Bit 31 ANMF	Accepted non-matching frame Acceptance of non-matching frames can be enabled via ANFS[1:0] and ANFE[1:0] bitfield of FDCAN_RXGFC. – 0: Received frame matching filter index FIDX – 1: Received frame did not match any Rx filter element
R1 Bits 30:24 FIDX[6:0]	Filter index 0-27=Index of matching Rx acceptance filter element (invalid if ANMF = 1). Range: 0 to LSS[4:0] - 1 or LSE[3:0] - 1 in FDCAN_RXGFC.
R1 Bit 21 FDF	FD format – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
R1 Bit 20 BRS	Bit rate switch – 0: Frame received without bit rate switching – 1: Frame received with bit rate switching
R1 Bits 19:16 DLC[3:0]	Data length code – 0-8: Classic CAN + CAN FD: received frame has 0-8 data bytes – 9-15: Classic CAN: received frame has 8 data bytes – 9-15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 Bits 15:0 RXTS[15:0]	Rx timestamp Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the timestamp counter prescaler TCP[3:0] of FDCAN_TSCC.
R2 Bits 31:24 DB3[7:0]	Data byte 3
R2 Bits 23:16 DB2[7:0]	Data byte 2
R2 Bits 15:8 DB1[7:0]	Data byte 1
R2 Bits 7:0 D[7:0]	Data byte 0
R3 Bits 31:24 DB7[7:0]	Data byte 7
R3 Bits 23:16 DB6[7:0]	Data byte 6

Table 569. Rx FIFO element description (continued)

Field	Description
R3 Bits 15:8 DB5[7:0]	Data byte 5
R3 Bits 7:0 DB4[7:0]	Data byte 4
⋮	⋮
Rn Bits 31:24 DBm[7:0]	Data byte m
Rn Bits 23:16 DBm-1[7:0]	Data byte m-1
Rn Bits 15:8 DBm-2[7:0]	Data byte m-2
Rn Bits 7:0 DBm-3[7:0]	Data byte m-3

49.3.9 FDCAN Tx buffer element

The Tx buffers section (three elements) can be configured to hold Tx FIFO or Tx queue. The Tx handler distinguishes between Tx FIFO and Tx queue using the Tx buffer configuration TFQM bit of the FDCAN_TXBC register. The element size is configured for storage of CAN FD messages with up to 64-byte data.

Table 570. Tx buffer and FIFO element

Bit	31 24	23 16	15 8	7 0
T0	ESI, XTD, RTR, ID[28:0]			
T1	MM[7:0]	EFC, Res., FDF, BRS, DLC[3:0]	Res.	
T2	DB3[7:0]	DB2[7:0]	DB1[7:0]	D[7:0]
T3	DB7[7:0]	DB6[7:0]	DB5[7:0]	DB4[7:0]
⋮	⋮	⋮	⋮	
Tn	DBm[7:0]	DBm-1[7:0]	DBm-2[7:0]	DBm-3[7:0]

Table 571. Tx buffer element description

Field	Description
T0 Bit 31 ESI(1)	Error state indicator – 0: ESI bit in CAN FD format depends only on error passive flag – 1: ESI bit in CAN FD format transmitted recessive
T0 Bit 30 XTD	Extended identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier

Table 571. Tx buffer element description (continued)

Field	Description
T0 Bit 29 RTR(2)	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
T0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 Bits 31:24 MM[7:0]	Message marker Written by CPU during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status.
T1 Bit 23 EFC	Event FIFO control – 0: Do not store Tx events – 1: Store Tx events
T1 Bit 21 FDF	FD format – 0: Frame transmitted in classic CAN format – 1: Frame transmitted in CAN FD format
T1 Bit 20 BRS(3)	Bit rate switching – 0: CAN FD frames transmitted without bit rate switching – 1: CAN FD frames transmitted with bit rate switching
T1 Bits 19:16 DLC[3:0]	Data length code – 0 - 8: Classic CAN + CAN FD: received frame has 0-8 data bytes – 9 - 15: Classic CAN: received frame has 8 data bytes – 9 - 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
T2 Bits 31:24 DB3[7:0]	Data byte 3
T2 Bits 23:16 DB2[7:0]	Data byte 2
T2 Bits 15:8 DB1[7:0]	Data byte 1
T2 Bits 7:0 D[7:0]	Data byte 0
T3 Bits 31:24 DB7[7:0]	Data byte 7
T3 Bits 23:16 DB6[7:0]	Data byte 6
T3 Bits 15:8 DB5[7:0]	Data byte 5
T3 Bits 7:0 DB4[7:0]	Data byte 4
⋮	⋮

Table 571. Tx buffer element description (continued)

Field	Description
Tn Bits 31:24 DBm[7:0]	Data byte m
Tn Bits 23:16 DBm-1[7:0]	Data byte m-1
Tn Bits 15:8 DBm-2[7:0]	Data byte m-2
Tn Bits 7:0 DBm-3[7:0]	Data byte m-3

1. The ESI bit of the transmit buffer is OR-ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node can optionally transmit the ESI bit recessive, but an error passive node always transmits the ESI bit recessive.
2. When RTR = 1, the FDCAN transmits a remote frame according to ISO11898-1, even if the transmission in CAN FD format is enabled by the FDOE bit of the FDCAN_CCCR.
3. Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled by setting the FDOE bit of the FDCAN_CCCR. Bit BRS is only evaluated when in addition BRSE bit is set in FDCAN_CCCR.

49.3.10 FDCAN Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx event, FIFO the host CPU gets this information in the order that the messages were transmitted. Status information about the Tx event FIFO can be obtained from FDCAN_TXEFS register.

Table 572. Tx event FIFO element

Bit	31			24	23						16	15		8	7		0
E0	ESI	XTD	RTR	ID[28:0]													
E1	MM[7:0]				ET[1:0]		EDL	BRS	DLC[3:0]			TXTS[15:0]					

Table 573. Tx event FIFO element description

Field	Description
E0 Bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
E0 Bit 30 XTD	Extended identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier
E0 Bit 29 RTR	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
E0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

Table 573. Tx event FIFO element description (continued)

Field	Description
E1 Bits 31:24 MM[7:0]	Message marker Copied from Tx buffer into Tx event FIFO element for identification of Tx message status.
E1 Bits 23:22 EFC	Event type – 00: Reserved – 01: Tx event – 10: Transmission in spite of cancellation (always set for transmissions in DAR mode) – 11: Reserved
E1 Bit 21 EDL	Extended data length – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
E1 Bit 20 BRS	Bit rate switching – 0: Frame transmitted without bit rate switching – 1: Frame transmitted with bit rate switching
T1 Bits 19:16 DLC[3:0]	Data length code 0 - 8: Frame with 0-8 data bytes transmitted 9 - 15: Frame with 8 data bytes transmitted
E1 Bits 15:0 TXTS[15:0]	Tx Timestamp Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the timestamp counter prescaler TCP[3:0] of FDCAN_TSCC.

49.3.11 FDCAN standard message ID filter element

Up to 28 filter elements can be configured for 11-bit standard IDs. When accessing a standard message ID filter element, its address is the filter list standard start address FLSSA plus the index of the filter element (0 … 27).

Table 574. Standard message ID filter element

Bit	31		24	23	16	15	8	7	0
S0	SFT[1:0]	SFEC[2:0]		SFID1[10:0]		Res.		SFID2[10:0]	

Table 575. Standard message ID filter element field description

Field	Description
Bit 31:30 SFT[1:0](1)	Standard filter type – 00: Range filter from SFID1 to SFID2 – 01: Dual ID filter for SFID1 or SFID2 – 10: Classic filter: SFID1 = filter, SFID2 = mask – 11: Filter element disabled
Bit 29:27 SFEC[2:0]	Standard filter element configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC[2:0] = 100, 101 or 110 a match sets interrupt flag IR.HPM and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Not used
Bits 26:16 SFID1[10:0]	Standard filter ID 1 First ID of standard ID filter element.
Bits 10:0 SFID2[10:0]	Standard filter ID 2 Second ID of standard ID filter element.

1. With SFT[1:0] = 11 the filter element is disabled and the acceptance filtering continues (same behavior as with SFEC[2:0] = 000).

Note: In case a reserved value is configured, the filter element is considered disabled.

49.3.12 FDCAN extended message ID filter element

Up to eight filter elements can be configured for 29-bit extended IDs. When accessing an extended message ID filter element, its address is the filter list extended start address FLESA plus twice the index of the filter element (0 … 7).

Table 576. Extended message ID filter element

Bit	31		24	23 16	15 8	7 0
F0	EFEC[2:0]		EFID1[28:0]			
F1	EFT[1:0]	Res.	EFID2[28:0]			

Table 577. Extended message ID filter element field description

Field	Description
F0 Bits 31:29 EFEC[2:0]	Extended filter element configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC[2:0] = 100, 101 or 110 a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Not used
F0 Bits 28:0 EFID1[28:0]	Extended filter ID 1 First ID of extended ID filter element. When filtering for Rx FIFO, this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism.
F1 Bits 31:30 EFT[1:0]	Extended filter type – 00: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID) – 01: Dual ID filter for EF1ID or EF2ID – 10: Classic filter: EF1ID = filter, EF2ID = mask – 11: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), XIDAM mask not applied
F1 Bit 29	Not used
F1 Bits 28:0 EFID2[28:0]	Extended filter ID 2 Second ID of extended ID filter element.

49.4 FDCAN registers

49.4.1 FDCAN core release register (FDCAN_CREL)

Address offset: 0x0000

Reset value: 0x3214 1218

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

Bits 31:28 **REL[3:0]**: 3

Bits 27:24 **STEP[3:0]**: 2

Bits 23:20 **SUBSTEP[3:0]**: 1

Bits 19:16 **YEAR[3:0]**: 4

Bits 15:8 **MON[7:0]**: 12

Bits 7:0 **DAY[7:0]**: 18

49.4.2 FDCAN endian register (FDCAN_ENDN)

Address offset: 0x0004

Reset value: 0x8765 4321

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

Bits 31:0 **ETV[31:0]**: Endianness test value

The endianness test value is 0x8765 4321.

Note: The register read must give the reset value to ensure no endianness issue.

49.4.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP)

Address offset: 0x000C

Reset value: 0x0000 0A33

This register is only writable if the CCE and INIT bits of the FDCAN_CCCR are set. The CAN time quantum can be programmed in the range of 1 to 32 FDCAN clock periods: t_q = (DBRP[4:0] + 1) FDCAN clock periods.

DTSEG1[4:0] is the sum of PROP_SEG and PHASE_SEG1. DTSEG2[3:0] is PHASE_SEG2. Therefore, the length of the bit time is
(programmed values) × [DTSEG1[4:0] + DTSEG2[3:0] + 3] × t_q or
(functional values) × [SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] × t_q.

The information processing time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]				
								rw			rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TDC**: Transceiver delay compensation
0: Transceiver delay compensation disabled
1: Transceiver delay compensation enabled

Bits 22:21 Reserved, must be kept at reset value.

Bits 20:16 **DBRP[4:0]**: Data bit rate prescaler
The value by which the oscillator frequency is divided to generate the bit time quanta. The bit time is built up from a multiple of this quantum. Valid values for the baud rate prescaler are 0 to 31. The hardware interpreters this value as the value programmed plus 1.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DTSEG1[4:0]**: Data time segment before sample point
Valid values are 0 to 31. The value used by the hardware is the one programmed, incremented by 1, that is t_{BS1} = (DTSEG1[4:0] + 1) × t_q.

Bits 7:4 **DTSEG2[3:0]**: Data time segment after sample point
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1, i.e. t_{BS2} = (DTSEG2[3:0] + 1) × t_q.

Bits 3:0 **DSJW[3:0]**: Synchronization jump width
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1: t_{SJW} = (DSJW[3:0] + 1) × t_q.

Note: With an FDCAN clock of 8 MHz, the reset value 0x0000 0A33 configures the FDCAN for a fast bit rate of 500 kbit/s.

The data phase bit rate must be higher than or equal to the nominal bit rate.

49.4.4 FDCAN test register (FDCAN_TEST)

Write access to this register is enabled by setting the TEST bit of the FDCAN_CCCR register. All register functions are set to their reset values when this bit is cleared.

Loop-back mode and software control of Tx pin FDCANx_TX are hardware test modes. Programming TX[1:0] differently from 00 can disturb the message transfer on the CAN bus.

Address offset: 0x0010

Reset value: 0x0000 0000

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

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

Bit 7 **RX**: Receive pin

This bit is used to monitor the actual value of FDCANx_RX. It is synchronized with the FDCANx_RX pin, so it is set after reset if the FDCAN is connected to a network.

0: The CAN bus is dominant (FDCANx_RX = 0)

1: The CAN bus is recessive (FDCANx_RX = 1)

Bits 6:5 **TX[1:0]**: Control of transmit pin

00: Reset value, FDCANx_TX TX is controlled by the CAN core, updated at the end of the CAN bit time

01: Sample point can be monitored at pin FDCANx_TX

10: Dominant (0) level at pin FDCANx_TX

11: Recessive (1) at pin FDCANx_TX

Bit 4 **LBCK**: Loop-back mode

0: Reset value, loop-back mode is disabled

1: Loop-back mode is enabled (see *Power-down (Sleep mode)*)

Bits 3:0 Reserved, must be kept at reset value.

49.4.5 FDCAN RAM watchdog register (FDCAN_RWD)

The RAM watchdog monitors the READY output of the message RAM. A message RAM access starts the message RAM watchdog counter with the value configured through the WDC[7:0] bitfield of the FDCAN_RWD register.

The counter is reloaded with WDC[7:0] when the message RAM signals successful completion by activating its READY output. In case there is no response from the message RAM until the counter has counted down to 0, the counter stops, and the interrupt flag WDI is set in the FDCAN_IR register. The RAM watchdog counter is clocked by the fdcan_pclk clock.

Address offset: 0x0014

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **WDV[7:0]**: Watchdog value
Actual message RAM watchdog counter value.

Bits 7:0 **WDC[7:0]**: Watchdog configuration
Start value of the message RAM watchdog counter. With the reset value of 00, the counter is disabled.
This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

49.4.6 FDCAN CC control register (FDCAN_CCCR)

Address offset: 0x0018

Reset value: 0x0000 0001

For details about setting and clearing single bits, see *Software initialization*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	r	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NISO**: Non-ISO operation
If this bit is set, the FDCAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.
0: CAN FD frame format according to ISO11898-1
1: CAN FD frame format according to Bosch CAN FD Specification V1.0

Bit 14 **TXP:** Transmit pause enable
If this bit is set, the FDCAN pauses for two CAN bit times before starting the next transmission after successfully transmitting a frame.
0: Disabled
1: Enabled

Bit 13 **EFBI**: Edge filtering during bus integration
0: Edge filtering disabled
1: Two consecutive dominant t_q required to detect an edge for hard synchronization

Bit 12 **PXHD**: Protocol exception handling disable
0: Protocol exception handling enabled
1: Protocol exception handling disabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **BRSE**: FDCAN bit rate switching
0: Bit rate switching for transmissions disabled
1: Bit rate switching for transmissions enabled

Bit 8 **FDOE**: FD operation enable
0: FD operation disabled
1: FD operation enabled

Bit 7 **TEST**: Test mode enable
0: Normal operation, FDCAN_TEST holds reset values
1: Test mode, write access to FDCAN_TEST enabled

Bit 6 **DAR**: Disable automatic retransmission
0: Automatic retransmission of messages not transmitted successfully enabled
1: Automatic retransmission disabled

Bit 5 **MON**: Bus monitoring mode
This bit can only be set by software when both CCE and INIT are set. The bit can be cleared by the host at any time.
0: Bus monitoring mode disabled
1: Bus monitoring mode enabled

Bit 4 **CSR**: Clock stop request
0: No clock stop requested
1: Clock stop requested. When clock stop is requested, first INIT and then CSA is set after all pending transfer requests have been completed and the CAN bus is idle.

Bit 3 **CSA**: Clock stop acknowledge
0: No clock stop acknowledged
1: FDCAN can be set in power-down by stopping APB clock and kernel clock.

Bit 2 **ASM**: ASM restricted operation mode
The restricted operation mode is intended for applications that adapt themselves to different CAN bit rates. The application tests different bit rates and leaves the restricted operation mode after it has received a valid frame. In the optional restricted operation mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or over.load frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus-idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can only be set by software when both CCE and INIT are set. The bit can be cleared by the software at any time.
0: Normal CAN operation
1: Restricted operation mode active

Bit 1 **CCE**: Configuration change enable
0: The CPU has no write access to the protected configuration registers.
1: The CPU has write access to the protected configuration registers (while INIT set in FDCAN_CCCR).

Bit 0 **INIT**: Initialization
0: Normal operation
1: Initialization started

Note: Due to the synchronization mechanism between the two clock domains, there can be a delay until the value written to INIT can be read back. Therefore, the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

49.4.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP)

Address offset: 0x001C

Reset value: 0x0600 0A03

This register is only writable if the CCE and INIT bits of the FDCAN_CCCR register are both set. The CAN bit time can be programed in the range of 4 to 81 × t_q. The CAN time quantum can be programmed in the range of 1 to 1024 FDCAN kernel clock periods:
t_q = (BRP + 1) × FDCAN clock period fdcan_ker_ck.

NTSEG1[7:0] is the sum of PROP_SEG and PHASE_SEG1. NTSEG2[6:0] is PHASE_SEG2. Therefore, the length of the bit time is
(programmed values) × [NTSEG1[7:0] + NTSEG2[6:0] + 3] × t_q or
(functional values) × [SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] × t_q.

The information processing time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSJW[6:0]							NBRP[8:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NTSEG1[7:0]								Res.	NTSEG2[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 **NSJW[6:0]**: Nominal (re)synchronization jump width

Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that the used value is the one programmed incremented by one.

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 24:16 **NBRP[8:0]**: Bit rate prescaler

Value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quantum. Valid values are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:8 **NTSEG1[7:0]**: Nominal time segment before sample point

Valid values are 0 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

This bitfield is write-protected write (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **NTSEG2[6:0]**: Nominal time segment after sample point

Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

Note: With a CAN kernel clock of 8 MHz, the reset value 0x0600 0A03 configures the FDCAN for a bit rate of 500 kBit/s.

49.4.8 FDCAN timestamp counter configuration register (FDCAN_TSCC)

Address offset: 0x0020

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **TCP[3:0]**: Timestamp counter prescaler

Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1…16].

The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

In CAN FD mode, the internal timestamp counter TCP does not provide a constant time base due to the different CAN bit times between arbitration phase and data phase. Thus CAN FD requires an external counter for timestamp generation (TSS[1:0] = 10).

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

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

Bits 1:0 **TSS[1:0]**: Timestamp select

00: Timestamp counter value always 0x0000

01: Timestamp counter value incremented according to TCP

10: External timestamp counter from TIM3 value (tim3_cnt[15:0])

11: Same as 00.

These bits are write-protected write (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

49.4.9 FDCAN timestamp counter value register (FDCAN_TSCV)

Address offset: 0x0024

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSC[15:0]**: Timestamp counter

The internal/external timestamp counter value is captured on start of frame (both Rx and Tx). When TSS[1:0] = 01 in FDCAN_TSCC, the timestamp counter is incremented in multiples of CAN bit times [1 … 16] depending on the configuration of TCP[3:0] in FDCAN_TSCC. A wrap around sets the TSW interrupt flag in FDCAN_IR. Write access resets the counter to 0.
When TSS[1:0] = 10, TSC[15:0] reflects the external timestamp counter value. A write access has no impact.

Note: A "wrap around" is a change of the timestamp counter value from non-0 to 0 that is not caused by write access to FDCAN_TSCV.

49.4.10 FDCAN timeout counter configuration register (FDCAN_TOCC)

Address offset: 0x0028

Reset value: 0xFFFF 0000

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

Bits 31:16 **TOP[15:0]**: Timeout period

Start value of the timeout counter (down-counter). Configures the timeout period.
This bitfield is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **TOS[1:0]**: Timeout select

When operating in continuous mode, a write to FDCAN_TOCV presets the counter to the value configured by TOP[15:0] in FDCAN_TOCC and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP[15:0]. Down-counting is started when the first FIFO element is stored.
00: Continuous operation
01: Timeout controlled by Tx event FIFO
10: Timeout controlled by Rx FIFO 0
11: Timeout controlled by Rx FIFO 1
This bitfield is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 0 **ETOC**: Timeout counter enable

0: Timeout counter disabled
1: Timeout counter enabled
This bit is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

For more details, see *Timeout counter*.

49.4.11 FDCAN timeout counter value register (FDCAN_TOCV)

Address offset: 0x002C

Reset value: 0x0000 FFFF

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TOC[15:0]**: Timeout counter

The timeout counter is decremented in multiples of CAN bit times [1 … 16] depending on the configuration of the TCP[3:0] bitfield of the FDCAN_TSCC register. When decremented to 0, the TOO interrupt flag is set in FDCAN_IR and the timeout counter is stopped. Start and reset/restart conditions are configured via TOS[1:0] in FDCAN_TOCC.

49.4.12 FDCAN error counter register (FDCAN_ECR)

Address offset: 0x0040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]							
								rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP	REC[6:0]							TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **CEL[7:0]**: CAN error logging

The counter is incremented each time when a CAN protocol error causes the transmit error counter or the receive error counter to be incremented. It is reset by read access to CEL[7:0]. The counter stops at 0xFF; the next increment of TEC[7:0] or REC[6:0] sets the ELO interrupt flag in FDCAN_IR.

Access type is rc_r: cleared on read.

Bit 15 **RP**: Receive error passive

0: The receive error counter is below the error passive level of 128.

1: The receive error counter has reached the error passive level of 128.

Bits 14:8 **REC[6:0]:** Receive error counter

Actual state of the receive error counter, values between 0 and 127.

Bits 7:0 **TEC[7:0]**: Transmit error counter

Actual state of the transmit error counter, values between 0 and 255.

When the ASM bit of the FDCAN_CCCR is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL[7:0] is still incremented.

49.4.13 FDCAN protocol status register (FDCAN_PSR)

Address offset: 0x0044

Reset value: 0x0000 0707

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PXE	REDL	RBRS	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	rc_r	rc_r	rc_r	rc_r	rs	rs	rs	r	r	r	r	r	rs	rs	rs

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

Bits 22:16 **TDCV[6:0]**: Transmitter delay compensation value

Position of the secondary sample point, defined by the sum of the measured delay from FDCAN_TX to FDCAN_RX and TDCO[6:0] in FDCAN_TDCR. The SSP position is, in the data phase, the number of minimum time quanta (mt_q) between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 × mt_q.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **PXE**: Protocol exception event

0: No protocol exception event occurred since last read access

1: Protocol exception event occurred

Bit 13 **REDL**: Received FDCAN message

This bit is set independent of acceptance filtering.

0: Since this bit was cleared by the CPU, no FDCAN message has been received.

1: Message in FDCAN format with EDL flag set has been received.

Access type is rc_r: cleared on read.

Bit 12 **RBRS:** BRS flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.

0: Last received FDCAN message did not have its BRS flag set.

1: Last received FDCAN message had its BRS flag set.

Access type is rc_r: cleared on read.

Bit 11 **RESI**: ESI flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.

0: Last received FDCAN message did not have its ESI flag set.

1: Last received FDCAN message had its ESI flag set.

Access type is rc_r: cleared on read.

Bits 10:8 **DLEC[2:0]**: Data last error code

Type of last error that occurred in the data phase of a FDCAN format frame with its BRS flag set. Coding is the same as for LEC[2:0]. This field is cleared when a FDCAN format frame with its BRS flag set has been transferred (reception or transmission) without error.

Access type is rs: set on read.

Bit 7 **BO**: Bus-off status

0: The FDCAN is not in bus-off state.

1: The FDCAN is in bus-off state.

Bit 6 **EW**: Warning status

0: Both error counters are below the error-warning limit of 96.

1: At least one of error counter has reached the error-warning limit of 96.

Bit 5 **EP**: Error passive

0: The FDCAN is in the error-active state. It normally takes part in bus communication and sends an active error flag when an error has been detected.

1: The FDCAN is in the error-passive state.

Bits 4:3 **ACT[1:0]**: Activity

Monitors the module's CAN communication state.

00: Synchronizing: node is synchronizing on CAN communication.

01: Idle: node is neither receiver nor transmitter.

10: Receiver: node is operating as receiver.

11: Transmitter: node is operating as transmitter.

Bits 2:0 **LEC[2:0]:** Last error code

LEC[2:0] indicates the type of the last error to occur on the CAN bus. This bitfield is cleared when a message has been transferred (reception or transmission) without error.

000: No error occurred since LEC[2:0] has been cleared by successful reception or transmission.

001: Stuff error. More than five equal bits in a sequence have occurred in a part of a received message where this is not allowed.

010: Form error. A fixed format part of a received frame has the wrong format.

011: Ack error. The message transmitted by the FDCAN was not acknowledged by another node.

100: Bit1 error. During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.

101: Bit0 error. During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During bus-off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the bus-off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).

110: CRC error. The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.

111: No change. Any read access to the protocol status register reinitializes LEC[2:0] to 7. When the LEC[2:0] shows the value 7, no CAN bus event was detected since the last CPU read access to the protocol status register.

Access type is rs: set on read.

Note: When a frame in FDCAN format has reached the data phase with the BRS flag set, the next CAN event (error or valid frame) is shown in DLEC[2:0] instead of LEC[2:0]. An error in a fixed stuff bit of an FDCAN CRC sequence is shown as a form error, not as a stuff error.

The bus-off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or clearing the INIT bit of the FDCAN_CCCR register. If the device enters bus-off, it sets the INIT bit of its own, stopping all bus activities. Once INIT has been cleared by the CPU, the device waits for 129 occurrences of bus-idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the bus-off recovery sequence, the error management counters are reset. During the waiting time after clearing INIT, each time a sequence of 11 recessive bits has been monitored, a bit0 error code is written to LEC[2:0] of FDCAN_PSR, enabling the CPU to check up whether the CAN bus is

stuck at dominant or continuously disturbed, and to monitor the bus-off recovery sequence. The REC[6:0] bitfield of the FDCAN_ECR register is used to count these sequences.

49.4.14 FDCAN transmitter delay compensation register (FDCAN_TDCR)

Address offset: 0x0048

Reset value: 0x0000 0000

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

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **TDCO[6:0]**: Transmitter delay compensation offset

Offset value defining the distance between the measured delay from FDCAN_TX to FDCAN_RX and the secondary sample point. Valid values are 0 to 127 × mt_q.

This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **TDCF[6:0]**: Transmitter delay compensation filter window length

Defines the minimum value for the SSP position, dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurements.

This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

49.4.15 FDCAN interrupt register (FDCAN_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the host clears them. A flag is cleared by writing 1 to the corresponding bit position.

Writing 0 has no effect. A hard reset clears the register. The configuration of FDCAN_IE controls whether an interrupt is generated. The configuration of FDCAN_ILS controls on which interrupt line an interrupt is signaled.

Address offset: 0x0050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOO	MRAF	TSW	TEFL	TEFF	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1N	RF0L	RF0F	RF0N
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ARA**: Access to reserved address
0: No access to reserved address occurred
1: Access to reserved address occurred

Bit 22 **PED**: Protocol error in data phase (data bit time is used)
0: No protocol error in data phase
1: Protocol error in data phase detected (DLEC[2:0] different from 0 and 7 in FDCAN_PSR)

Bit 21 **PEA**: Protocol error in arbitration phase (nominal bit time is used)
0: No protocol error in arbitration phase
1: Protocol error in arbitration phase detected (LEC[2:0] different from 0 and 7 in FDCAN_PSR)

Bit 20 **WDI**: Watchdog interrupt
0: No message RAM watchdog event occurred
1: Message RAM watchdog event due to missing READY

Bit 19 **BO**: Bus-off status
0: Bus-off status unchanged
1: Bus-off status changed

Bit 18 **EW**: Warning status
0: Error-warning status unchanged
1: Error-warning status changed

Bit 17 **EP**: Error passive
0: Error-passive status unchanged
1: Error-passive status changed

Bit 16 **ELO**: Error logging overflow
0: CAN error logging counter did not overflow.
1: Overflow of CAN error logging counter occurred.

Bit 15 **TOO**: Timeout occurred
0: No timeout
1: Timeout reached

Bit 14 **MRAF**: Message RAM access failure
The flag is set when the Rx handler:
- has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx handler starts processing of the following message.
- was unable to write a message to the message RAM. In this case message storage is aborted.

In both cases the FIFO put index is not updated. The partly stored message is overwritten when the next message is stored to this location.
The flag is also set when the Tx handler was not able to read a message from the message RAM in time. In this case message transmission is aborted. In case of a Tx handler access failure, the FDCAN is switched into restricted operation mode (see *Restricted operation mode*). To leave restricted operation mode, the host CPU has to clear the ASM of the FDCAN_CCCR register.
0: No message RAM access failure occurred
1: Message RAM access failure occurred

Bit 13 **TSW**: Timestamp wraparound
0: No timestamp counter wrap-around
1: Timestamp counter wrapped around

Bit 12 **TEFL**: Tx event FIFO element lost
0: No Tx event FIFO element lost
1: Tx event FIFO element lost

Bit 11 **TEFF**: Tx event FIFO full
0: Tx event FIFO Not full
1: Tx event FIFO full

Bit 10 **TEFN**: Tx event FIFO new entry
0: Tx event FIFO unchanged
1: Tx handler wrote Tx event FIFO element.

Bit 9 **TFE**: Tx FIFO empty
0: Tx FIFO non-empty
1: Tx FIFO empty

Bit 8 **TCF**: Transmission cancellation finished
0: No transmission cancellation finished
1: Transmission cancellation finished

Bit 7 **TC**: Transmission completed
0: No transmission completed
1: Transmission completed

Bit 6 **HPM**: High-priority message
0: No high-priority message received
1: High-priority message received

Bit 5 **RF1L**: Rx FIFO 1 message lost
0: No Rx FIFO 1 message lost
1: Rx FIFO 1 message lost

Bit 4 **RF1F**: Rx FIFO 1 full
0: Rx FIFO 1 not full
1: Rx FIFO 1 full

Bit 3 **RF1N**: Rx FIFO 1 new message
0: No new message written to Rx FIFO 1
1: New message written to Rx FIFO 1

Bit 2 **RF0L**: Rx FIFO 0 message lost
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost

Bit 1 **RF0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bit 0 **RF0N**: Rx FIFO 0 new message
0: No new message written to Rx FIFO 0
1: New message written to Rx FIFO 0

49.4.16 FDCAN interrupt enable register (FDCAN_IE)

The settings in the interrupt enable register determine which status changes in the interrupt register are signaled on an interrupt line.

Address offset: 0x0054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1NE	RF0LE	RF0FE	RF0NE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ARAE**: Access to reserved address enable

Bit 22 **PEDE**: Protocol error in data phase enable

Bit 21 **PEAE**: Protocol error in arbitration phase enable

Bit 20 **WDIE**: Watchdog interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 19 **BOE**: Bus-off status
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **EWE**: Warning status interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **EPE**: Error passive interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **ELOE**: Error logging overflow interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 15 **TOOE**: Timeout occurred interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 14 **MRAFE:** Message RAM access failure interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **TSWE**: Timestamp wraparound interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **TEFLE**: Tx event FIFO element lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **TEFFE**: Tx event FIFO full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **TEFNE**: Tx event FIFO new entry interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 9 **TFEE**: Tx FIFO empty interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 8 **TCFE**: Transmission cancellation finished interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 7 **TCE**: Transmission completed interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 6 **HPME**: High-priority message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 5 **RF1LE**: Rx FIFO 1 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 4 **RF1FE**: Rx FIFO 1 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 3 **RF1NE**: Rx FIFO 1 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 2 **RF0LE**: Rx FIFO 0 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 1 **RF0FE**: Rx FIFO 0 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **RF0NE**: Rx FIFO 0 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

49.4.17 FDCAN interrupt line select register (FDCAN_ILS)

This register assigns an interrupt generated by a specific group of interrupt flags from the interrupt register to one of the two module interrupt lines. For interrupt generation, the respective interrupt line must be enabled via the EINT0 and EINT1 bit of the FDCAN_ILE register. 0 indicates that the interrupt is assigned to interrupt line fdcan1_intr0_it, 1 indicates that the interrupt is assigned to interrupt line fdcan1_intr1_it.

Address offset: 0x0058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERR	BERR	MISC	TFERR	SMSG	RXFIFO1	RXFIFO0
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **PERR:** Protocol error grouping the following interruption
- ARAL: Access to reserved address line
- PEDL: Protocol error in data phase line
- PEAL: Protocol error in arbitration phase line
- WDIL: Watchdog interrupt line
- BOL: Bus-off status
- EWL: Warning status interrupt line

Bit 5 **BERR:** Bit and line error grouping the following interruption
- EPL Error passive interrupt line
- ELOL: Error logging overflow interrupt line

Bit 4 **MISC:** Interrupt regrouping the following interruption
- TOOL: Timeout occurred interrupt line
- MRAFL: Message RAM access failure interrupt line
- TSWL: Timestamp wraparound interrupt line

Bit 3 **TFERR:** Tx FIFO ERROR grouping the following interruption
- TEFLL: Tx event FIFO element lost interrupt line
- TEFFL: Tx event FIFO full interrupt line
- TEFNL: Tx event FIFO new entry interrupt line
- TFEL: Tx FIFO empty interrupt line

Bit 2 **SMSG:** Status message bit grouping the following interruption
- TCFL: Transmission cancellation finished interrupt line
- TCL: Transmission completed interrupt line
- HPML: High-priority message interrupt line

Bit 1 **RXFIFO1:** RX FIFO bit grouping the following interruption
- RF1LL: Rx FIFO 1 message lost interrupt line
- RF1FL: Rx FIFO 1 full interrupt line
- RF1NL: Rx FIFO 1 new message interrupt line

Bit 0 **RXFIFO0:** RX FIFO bit grouping the following interruption
RF0LL: Rx FIFO 0 message lost interrupt line
RF0FL: Rx FIFO 0 full interrupt line
RF0NL: Rx FIFO 0 new message interrupt line

49.4.18 FDCAN interrupt line enable register (FDCAN_ILE)

Each of the two interrupt lines to the CPU can be enabled/disabled separately by programming the EINT0 and EINT1 bits.

Address offset: 0x005C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
														rw	rw

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

Bit 1 **EINT1**: Enable interrupt line 1
0: Interrupt line fdcan_intr0_it disabled
1: Interrupt line fdcan_intr0_it enabled

Bit 0 **EINT0**: Enable interrupt line 0
0: Interrupt line fdcan_intr1_it disabled
1: Interrupt line fdcan_intr1_it enabled

49.4.19 FDCAN global filter configuration register (FDCAN_RXGFC)

Global settings for message ID filtering. The global filter configuration controls the filter path for standard and extended messages as described in *Figure 1008* and *Figure 1009*.

Address offset: 0x0080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	LSE[3:0]				Res.	Res.	Res.	LSS[4:0]				
				rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F0OM	F1OM	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
						rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **LSE[3:0]**: Number of extended filter elements in the list
0: No extended message ID filter
1 to 8: Number of extended message ID filter elements
> 8: Values greater than 8 are interpreted as 8.
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **LSS[4:0]**: Number of standard filter elements in the list
0: No standard message ID filter
1 to 28: Number of standard message ID filter elements
> 28: Values greater than 28 are interpreted as 28.
This bitfield is write protected (P), which means that write access by the bits is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **F0OM**: FIFO 0 operation mode (overwrite or blocking)
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 8 **F1OM**: FIFO 1 operation mode (overwrite or blocking)
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ANFS[1:0]**: Accept Non-matching frames standard
Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
This bitfield is write-protected (P), which means write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 3:2 **ANFE[1:0]**: Accept non-matching frames extended
Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 1 **RRFS**: Reject remote frames standard
0: Filter remote frames with 11-bit standard IDs
1: Reject all remote frames with 11-bit standard IDs
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 0 **RRFE**: Reject remote frames extended
0: Filter remote frames with 29-bit standard IDs
1: Reject all remote frames with 29-bit standard IDs
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

49.4.20 FDCAN extended ID and mask register (FDCAN_XIDAM)

Address offset: 0x0084

Reset value: 0x1FFF FFFF

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **EIDM[28:0]**: Extended ID mask
For acceptance filtering of extended frames the extended ID AND mask is AND-ed with the message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set, the mask is not active.
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

49.4.21 FDCAN high-priority message status register (FDCAN_HPMS)

This register is updated every time a message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address offset: 0x0088

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FLST**: Filter list
Indicates the filter list of the matching filter element:
0: Standard filter list
1: Extended filter list

Bits 14:13 Reserved, must be kept at reset value.

Bits 12:8 **FIDX[4:0]**: Filter index
Index of matching filter element.
Range: 0 to LSS[4:0] - 1 or LSE[3:0] - 1 in FDCAN_RXGFC.

Bits 7:6 **MSI[1:0]**: Message storage indicator
00: No FIFO selected
01: FIFO overrun
10: Message stored in FIFO 0
11: Message stored in FIFO 1

Bits 5:3 Reserved, must be kept at reset value.

Bits 2:0 **BIDX[2:0]**: Buffer index
Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

49.4.22 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S)

Address offset: 0x0090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RF0L	F0F	Res.	Res.	Res.	Res.	Res.	Res.	F0PI[1:0]	
						r	r							r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F0GI[1:0]		Res.	Res.	Res.	Res.	F0FL[3:0]			
						r	r					r	r	r	r

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

Bit 25 **RF0L**: Rx FIFO 0 message lost
This bit is a copy of the RF0L interrupt flag of the FDCAN_IR register. When RF0L is cleared, this bit is also cleared.
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size 0

Bit 24 **F0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F0PI[1:0]**: Rx FIFO 0 put index
Rx FIFO 0 write index pointer.
Range: 0 to 2.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **F0GI[1:0]**: Rx FIFO 0 get index
Rx FIFO 0 read index pointer.
Range: 0 to 2.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **F0FL[3:0]**: Rx FIFO 0 fill level
Number of elements stored in Rx FIFO 0.
Range: 0 to 3.

49.4.23 CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A)

Address offset: 0x0094

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[2:0]		
													rw	rw	rw

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

Bits 2:0 **F0AI[2:0]**: Rx FIFO 0 acknowledge index

After the host has read a message or a sequence of messages from Rx FIFO 0, it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI[2:0]. This sets the Rx FIFO 0 get index (F0GI[1:0] of FDCAN_RXF0S) to F0AI[2:0] + 1 and updates the FIFO 0 fill level (F0FL[3:0] FDCAN_RXF0S).

49.4.24 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S)

Address offset: 0x0098

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RF1L	F1F	Res.	Res.	Res.	Res.	Res.	Res.	F1PI[1:0]	
						r	r							r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F1GI[1:0]		Res.	Res.	Res.	Res.	F1FL[3:0]			
						r	r					r	r	r	r

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

Bit 25 **RF1L**: Rx FIFO 1 message lost

This bit is a copy of the RF1L interrupt flag of the FDCAN_IR register. When RF1L is cleared, this bit is also cleared.

0: No Rx FIFO 1 message lost

1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size 0

Bit 24 **F1F**: Rx FIFO 1 full

0: Rx FIFO 1 not full

1: Rx FIFO 1 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F1PI[1:0]**: Rx FIFO 1 put index

Rx FIFO 1 write index pointer.

Range: 0 to 2.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **F1GI[1:0]**: Rx FIFO 1 get index
Rx FIFO 1 read index pointer.
Range: 0 to 2.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **F1FL[3:0]**: Rx FIFO 1 fill level
Number of elements stored in Rx FIFO 1.
Range: 0 to 3.

49.4.25 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)

Address offset: 0x009C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[2:0]		
													rw	rw	rw

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

Bits 2:0 **F1AI[2:0]**: Rx FIFO 1 acknowledge index
After the host has read a message or a sequence of messages from Rx FIFO 1, it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI[2:0]. This sets the Rx FIFO 1 get index (F1GI[1:0] of FDCAN_RXF1S) to F1AI[2:0] + 1 and updates the FIFO 1 fill level (F1FL[3:0] FDCAN_RXF1S).

49.4.26 FDCAN Tx buffer configuration register (FDCAN_TXBC)

Address offset: 0x00C0

Reset value: 0x0000 0000

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

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TFQM**: Tx FIFO/queue mode
0: Tx FIFO operation
1: Tx queue operation.
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

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

49.4.27 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS)

The Tx FIFO/queue status is related to the pending Tx requests listed in the FDCAN_TXBRP register. Therefore, the effect of add/cancellation requests can be delayed due to a running Tx scan (FDCAN_TXBRP not yet updated).

Address offset: 0x00C4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	Res.	Res.	Res.	TFQPI[1:0]	
										r				r	r

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

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

Bit 21 **TFQF**: Tx FIFO/queue full
0: Tx FIFO/queue not full
1: Tx FIFO/queue full

Bits 20:18 Reserved, must be kept at reset value.

Bits 17:16 **TFQPI[1:0]**: Tx FIFO/queue put index
Tx FIFO/queue write index pointer, range 0 to 3

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **TFGI[1:0]**: Tx FIFO get index
Tx FIFO read index pointer, range 0 to 3. Read as 0 when Tx queue operation is configured (TFQM = 1 in FDCAN_TXBC)

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

Bits 2:0 **TFFL[2:0]**: Tx FIFO free level
Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 3. Read as 0 when Tx queue operation is configured (TFQM = 1 in FDCAN_TXBC).

49.4.28 FDCAN Tx buffer request pending register (FDCAN_TXBRP)

Address offset: 0x00C8

Reset value: 0x0000 0000

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

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

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

Bits 2:0 **TRP[2:0]**: Transmission request pending

Each Tx buffer has its own transmission request pending bit. The bits are set via the FDCAN_TXBAR register. The bits are cleared after a requested transmission has completed or has been canceled via the FDCAN_TXBCR register.

After the FDCAN_TXBRP bit has been set, a Tx scan is started to check for the pending Tx request with the highest priority (Tx buffer with lowest message ID).

A cancellation request resets the corresponding transmission request pending bit of the FDCAN_TXBRP register. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are directly cleared after the corresponding FDCAN_TXBRP bit has been cleared.

After a cancellation has been requested, a finished cancellation is signaled via the FDCAN_TXBCF in the following cases:

- after successful transmission together with the corresponding TXBTO bit
- when the transmission has not yet been started at the point of cancellation
- when the transmission has been aborted due to lost arbitration
- when an error occurred during frame transmission

In DAR mode, all transmissions are automatically canceled if they are not successful. The corresponding FDCAN_TXBCF bit is set for all unsuccessful transmissions.

0: No transmission request pending

1: Transmission request pending

Note: FDCAN_TXBRP bits set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this add request is canceled immediately. The corresponding FDCAN_TXBRP bit is cleared.

49.4.29 FDCAN Tx buffer add request register (FDCAN_TXBAR)

Address offset: 0x00CC

Reset value: 0x0000 0000

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

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

Bits 2:0 **AR[2:0]**: Add request

Each Tx buffer has its own add request bit. Writing a 1 sets the corresponding add request bit; writing a 0 has no impact. This enables the host to set transmission requests for multiple Tx buffers with one write to FDCAN_TXBAR. When no Tx scan is running, the bits are cleared immediately, else the bits remain set until the Tx scan process has completed.

0: No transmission request added

1: Transmission requested added.

Note: If an add request is applied for a Tx buffer with pending transmission request (corresponding FDCAN_TXBRP bit already set), the request is ignored.

49.4.30 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR)

Address offset: 0x00D0

Reset value: 0x0000 0000

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

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

Bits 2:0 **CR[2:0]**: Cancellation request

Each Tx buffer has its own cancellation request bit. Writing a 1 sets the corresponding CR bit; writing a 0 has no impact.

This enables the host to set cancellation requests for multiple Tx buffers with one write to FDCAN_TXBCR. The bits remain set until the corresponding FDCAN_TXBRP bit is cleared.

0: No cancellation pending

1: Cancellation pending

49.4.31 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO)

Address offset: 0x00D4

Reset value: 0x0000 0000

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

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

Bits 2:0 **TO[2:0]**: Transmission occurred.

Each Tx buffer has its own TO bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a successful transmission. The bits are cleared when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN_TXBAR.

0: No transmission occurred

1: Transmission occurred

49.4.32 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF)

Address offset: 0x00D8

Reset value: 0x0000 0000

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

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

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

Bits 2:0 **CF[2:0]**: Cancellation finished

Each Tx buffer has its own CF bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a cancellation was requested via FDCAN_TXBCR. In case the corresponding FDCAN_TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are cleared when a new transmission is requested by writing a 1 to the corresponding bit of the FDCAN_TXBAR register.

0: No transmit buffer cancellation

1: Transmit buffer cancellation finished

49.4.33 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE)

Address offset: 0x00DC

Reset value: 0x0000 0000

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

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

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

Bits 2:0 **TIE[2:0]**: Transmission interrupt enable

Each Tx buffer has its own TIE bit.

0: Transmission interrupt disabled

1: Transmission interrupt enable

49.4.34 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE)

Address offset: 0x00E0

Reset value: 0x0000 0000

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

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

Bits 2:0 **CFIE[2:0]**: Cancellation finished interrupt enable.
Each Tx buffer has its own CFIE bit.
0: Cancellation finished interrupt disabled
1: Cancellation finished interrupt enabled

49.4.35 FDCAN Tx event FIFO status register (FDCAN_TXEFS)

Address offset: 0x00E4

Reset value: 0x0000 0000

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

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

Bit 25 **TEFL**: Tx event FIFO element lost
This bit is a copy of the TEFL interrupt flag of the FDCAN_IR. When TEFL is cleared, this bit is also cleared.
0 No Tx event FIFO element lost
1 Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size 0.

Bit 24 **EFF**: Event FIFO full
0: Tx event FIFO not full
1: Tx event FIFO full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **EFPI[1:0]**: Event FIFO put index
Tx event FIFO write index pointer.
Range: 0 to 3.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **EFGI[1:0]**: Event FIFO get index
Tx event FIFO read index pointer.
Range: 0 to 3.

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

Bits 2:0 **EFFL[2:0]**: Event FIFO fill level
Number of elements stored in Tx event FIFO.
Range: 0 to 3.

49.4.36 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA)

Address offset: 0x00E8

Reset value: 0x0000 0000

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

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

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

Bits 1:0 **EFAI[1:0]**: Event FIFO acknowledge index
After the host has read an element or a sequence of elements from the Tx event FIFO, it has to write the index of the last element read from Tx event FIFO to EFAI[1:0]. This sets the Tx event FIFO get index (EFGI[1:0] of FDCAN_TXEFS) to EFAI[1:0] + 1 and updates the FIFO 0 fill level (EFFL[2:0] of FDCAN_TXEFS).

49.4.37 FDCAN CFG clock divider register (FDCAN_CKDIV)

Address offset: 0x0100

Reset value: 0x0000 0000

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

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

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

Bits 3:0 **PDIV[3:0]**: input clock divider

The CAN kernel clock can be divided prior to be used by the CAN subsystem. The rate must be computed using the divider output clock.

0000: Divide by 1
0001: Divide by 2
0010: Divide by 4
0011: Divide by 6
0100: Divide by 8
0101: Divide by 10
0110: Divide by 12
0111: Divide by 14
1000: Divide by 16
1001: Divide by 18
1010: Divide by 20
1011: Divide by 22
1100: Divide by 24
1101: Divide by 26
1110: Divide by 28
1111: Divide by 30

This bitfield is write-protected (P): which means that write access is possible only when the CCE bit of the FDCAN_CCCR register is set.

Note: The clock divider is common to all FDCAN instances. Only FDCAN1 instance has FDCAN_CKDIV register, which changes clock divider for all instances.

If several FDCAN instances are present, the input clock divider must be modified before configuring the other FDCAN instances.

49.4.38 FDCAN register map

Table 578. FDCAN register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	FDCAN_CREL	REL[3:0]				STEP[3:0]				SUBSTEP [3:0]				YEAR[3:0]				MON[7:0]								DAY[7:0]							
	Reset value	0	0	0	0	0	0	0	1	1	1	1	0	1	0	1	0	0	1	1	0	1	1	1	1	1	0	1	0	0	0	1	0
0x0004	FDCAN_ENDN	ETV[31:0]																															
	Reset value	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1
0x0008	Reserved	Reserved																															
0x000C	FDCAN_DBTP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]					Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
	Reset value									0	0	0	0	0	0	0	0				0	1	0	1	0	0	0	1	1	0	0	1	1
0x0010	FDCAN_TEST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX[1:0]		LBCK	Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				

Table 578. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0014	FDCAN_RWD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDV[7:0]								WDC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	FDCAN_CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INT
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x001C	FDCAN_NBTP	NSJW[6:0]							NBRP[8:0]									NTSEG1[7:0]								Res.	NTSEG2[6:0]						
	Reset value	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1
0x0020	FDCAN_TSCC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	FDCAN_TSCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	FDCAN_TOCC	TOP[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS[1:0]		ETOC
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	FDCAN_TOCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOC[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0030-0x003C	Reserved	Reserved																															
0x0040	FDCAN_ECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]								RP	REC[6:0]							TEC[7:0]							
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	FDCAN_PSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]							Res.	PXE	REDL	RBRSRESI1	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	Reset value									0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x0048	FDCAN_TDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCO[6:0]							Res.	TDCF[6:0]						
	Reset value																		0	0	0	0	0	0		0	0	0	0	0	0	0	0
0x004C	Reserved	Reserved																															
0x0050	FDCAN_IR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO	TOO	MRAF	TSW	TEFL	TEFF	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1N	RF0L	RF0F	RF0N
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054	FDCAN_IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1NE	RF0LE	RF0FE	RF0NE
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 578. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0058	FDCAN_ILS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERR	BERR	MISC	TFERR	SMSG	RXFIFO1	RXFIFO0
	Reset value	0																									0	0	0	0	0	0	0
0x005C	FDCAN_ILE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
	Reset value																															0	0
0x0060-0x007C	Reserved	Reserved																															
0x0080	FDCAN_RXGFC	Res.	Res.	Res.	Res.	LSE[3:0].				Res.	Res.	Res.	LSS[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	F0OM	F1OM	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
	Reset value					0	0	0	0				0	0	0	0	0							0	0			0	0	0	0	0	0
0x0084	FDCAN_XIDAM	Res.	Res.	Res.	EIDM[28:0]																												
	Reset value				1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0088	FDCAN_HPMS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLST	Res.	Res.	FIDX[4:0]						MSI[1:0]		Res.	Res.	Res.	BIDX[2:0]		
	Reset value																	0			0	0	0	0	0	0	0	0				0	0	0
0x0090	FDCAN_RXF0S	Res.	Res.	Res.	Res.	Res.	Res.	RF0L	F0F	Res.	Res.	Res.	Res.	Res.	Res.	F0PI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	F0GI[1:0]		Res.	Res.	Res.	Res.	Res.	F0FL[3:0]		
	Reset value							0	0							0	0							0	0						0	0	0	0
0x0094	FDCAN_RXF0A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[2:0]		
	Reset value																														0	0	0
0x0098	FDCAN_RXF1S	Res.	Res.	Res.	Res.	Res.	Res.	RF1L	F1F	Res.	Res.	Res.	Res.	Res.	Res.	F1PI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	F1GI[1:0]		Res.	Res.	Res.	Res.	Res.	F1FL[3:0]		
	Reset value							0	0							0	0							0	0						0	0	0	0
0x009C	FDCAN_RXF1A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[2:0]		
	Reset value																														0	0	0
0x00A0-0x00BC	Reserved	Reserved																															
0x00C0	FDCAN_TXBC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0	0																							
0x00C4	FDCAN_TXFQS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	Res.	Res.	Res.	TFQPI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	TFGI[1:0]		Res.	Res.	Res.	Res.	Res.	TFFL[2:0]		
	Reset value											0				0	0							0	0						0	1	1

Table 578. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00C8	FDCAN_TXBRP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRP2	TRP1	TRP0
	Reset value																														0	0	0
0x00CC	FDCAN_TXBAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR2	AR1	AR0
	Reset value																														0	0	0
0x00D0	FDCAN_TXBCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CR2	CR1	CR0
	Reset value																														0	0	0
0x00D4	FDCAN_TXBTO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TO2	TO1	TO0
	Reset value																														0	0	0
0x00D8	FDCAN_TXBCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CF2	CF1	CF0
	Reset value																														0	0	0
0x00DC	FDCAN_TXBTIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE2	TIE1	TIE0
	Reset value																														0	0	0
0x00E0	FDCAN_TXBCIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFIE2	CFIE1	CFIE0
	Reset value																														0	0	0
0x00E4	FDCAN_TXEFS	Res.	Res.	Res.	Res.	Res.	Res.	TEFL	EFF	Res.	Res.	Res.	Res.	Res.	Res.	EFPI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	EFG[1:0]		Res.	Res.	Res.	Res.	Res.	EFFL[2:0]		
	Reset value							0	0							0	0							0	0						0	0	0
0x00E8	FDCAN_TXEFA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI[1:0]	
	Reset value																															0	0
0x0100	FDCAN_CKDIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDIV[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

50 USB on-the-go full-speed (OTG_FS)

50.1 OTG_FS introduction

Portions Copyright (c) Synopsys, Inc. All rights reserved. Used with permission.

This section presents the architecture and the programming model of the OTG_FS controller.

The following acronyms are used throughout the section:

FS	Full-speed
LS	Low-speed
MAC	Media access controller
OTG	On-the-go
PFC	Packet FIFO controller
PHY	Physical layer
USB	Universal serial bus
UTMI	USB 2.0 Transceiver Macrocell interface (UTMI)
LPM	Link power management
BCD	Battery charging detector
HNP	Host negotiation protocol
SRP	Session request protocol

References are made to the following documents:

- USB On-The-Go Supplement, Revision 2.0
- Universal Serial Bus Revision 2.0 Specification
- USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007
- Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007
- Battery Charging Specification, Revision 1.2

The USB OTG is a dual-role device (DRD) controller that supports both device and host functions and is fully compliant with the *On-The-Go Supplement to the USB 2.0 Specification*. It can also be configured as a host-only or device-only controller, fully compliant with the *USB 2.0 Specification.* OTG_FS supports the speeds defined in the *Table 579: OTG_FS speeds supported* below. The only external device required is a charge pump for V_{BUS} in OTG mode.

Table 579. OTG_FS speeds supported

-	HS (480 Mb/s)	FS (12 Mb/s)	LS (1.5 Mb/s)
Host mode	-	X	X
Device mode	-	X	-

50.2 OTG_FS main features

The main features can be divided into three categories: general, host-mode and device-mode features.

50.2.1 General features

The OTG_FS interface general features are the following:

- It is USB-IF certified to the Universal Serial Bus Specification Rev 2.0
- OTG_FS supports the following PHY interface:
  - An on-chip full-speed PHY
- It includes full support (PHY) for the optional On-The-Go (OTG) protocol detailed in the On-The-Go Supplement Rev 2.0 specification
  - Integrated support for A-B device identification (ID line)
  - It allows host to turn V_{BUS} off to conserve battery power in OTG applications
  - It supports OTG monitoring of V_{BUS} levels with internal comparators
  - It supports dynamic host-peripheral switch of role
- It is software-configurable to operate as:
  - USB On-The-Go Full-Speed Dual Role device
- It supports FS SOF and LS Keep-alives with
  - SOF pulse PAD connectivity
  - SOF pulse internal connection to timer (TIMx)
  - Configurable framing period
  - Configurable end of frame interrupt
- It includes power saving features such as system stop during USB suspend, switch-off of clock domains internal to the digital core, PHY and DFIFO power management.
- It features a dedicated RAM of 1.25 Kbytes with advanced FIFO control:
  - Configurable partitioning of RAM space into different FIFOs for flexible and efficient use of RAM
  - Each FIFO can hold multiple packets
  - Dynamic memory allocation
  - Configurable FIFO sizes that are not powers of 2 to allow the use of contiguous memory locations
- It guarantees max USB bandwidth for up to one frame (1 ms) without system intervention.
- It supports charging port detection as described in Battery Charging Specification Revision 1.2.

50.2.2 Host-mode features

The OTG_FS interface main features and requirements in host-mode are the following:

- External charge pump for V_{BUS} voltage generation.
- Up to 12 host channels (pipes): each channel is dynamically reconfigurable to allocate any type of USB transfer.
- Built-in hardware scheduler holding:
  - Up to 12 interrupt plus isochronous transfer requests in the periodic hardware queue
  - Up to 12 control plus bulk transfer requests in the non-periodic hardware queue
- Management of a shared Rx FIFO, a periodic Tx FIFO and a nonperiodic Tx FIFO for efficient usage of the USB data RAM.

50.2.3 Peripheral-mode features

The OTG_FS interface main features in peripheral-mode are the following:

- 1 bidirectional control endpoint0
- 5 IN endpoints (EPs) configurable to support bulk, interrupt or isochronous transfers
- 5 OUT endpoints configurable to support bulk, interrupt or isochronous transfers
- Management of a shared Rx FIFO and a Tx-OUT FIFO for efficient usage of the USB data RAM
- Management of up to 6 dedicated Tx-IN FIFOs (one for each active IN EP) to put less load on the application
- Support for the soft disconnect feature.

50.3 OTG_FS implementation

Table 580. OTG_FS implementation(1)

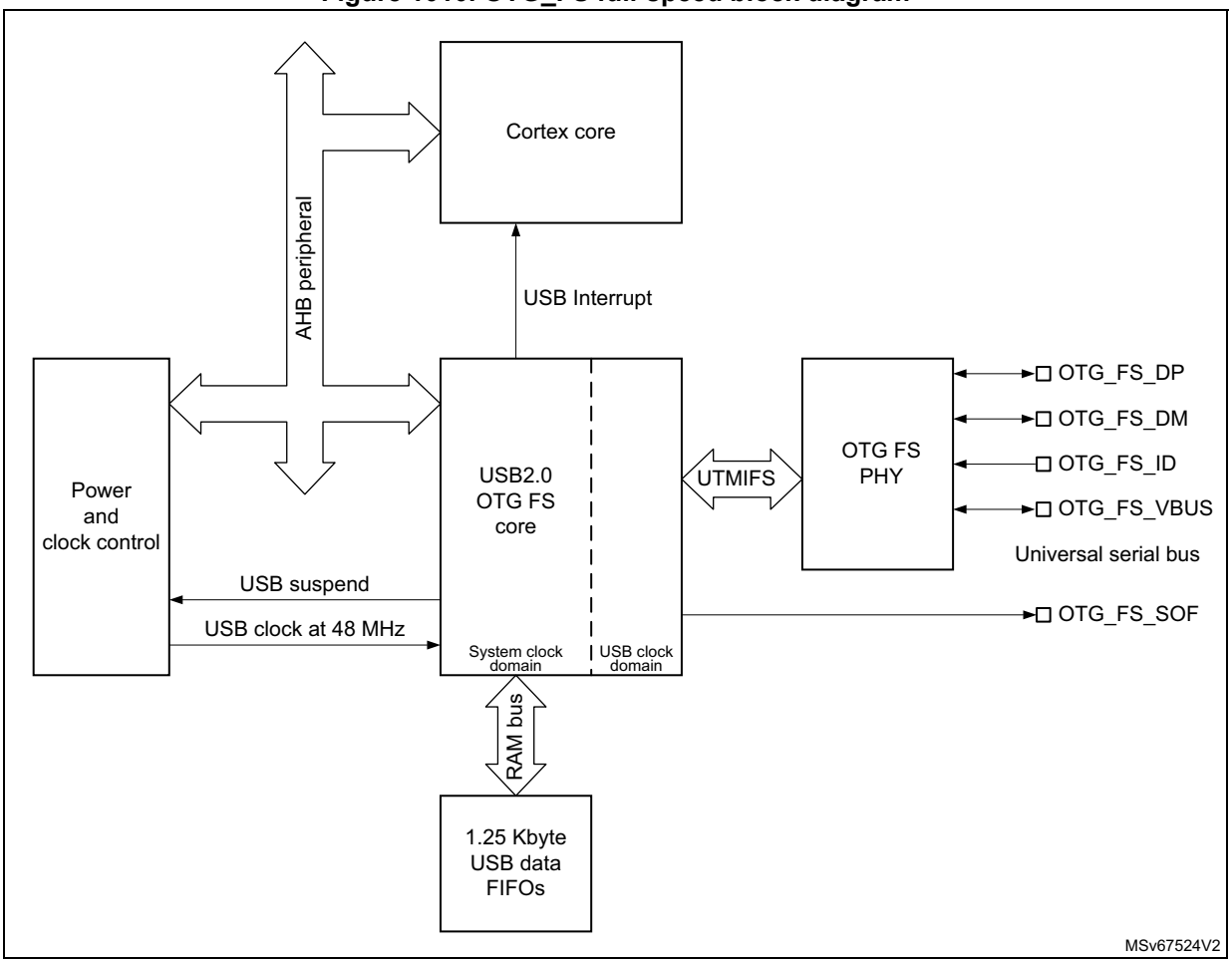
USB features	OTG_FS
Device bidirectional endpoints (including EP0)	6
Host mode channels	12
Size of dedicated SRAM	1.2 Kbytes
USB 2.0 link power management (LPM) support	X
OTG revision supported	2.0
Battery charging detection (BCD) support	X
Integrated PHY	FS

1. “X” = supported, “-” = not supported, “FS” = supported in FS mode, “HS” = supported in HS mode.

50.4 OTG_FS functional description

50.4.1 OTG_FS block diagram

Figure 1010. OTG_FS full-speed block diagram



50.4.2 OTG_FS pin and internal signals

Table 581. OTG_FS input/output pins

Signal name	Signal type	Description
OTG_FS_DP	Digital input/output	USB OTG D+ line
OTG_FS_DM	Digital input/output	USB OTG D- line
OTG_FS_ID	Digital input	USB OTG ID
OTG_FS_VBUS	Analog input	USB OTG VBUS
OTG_FS_SOF	Digital output	USB OTG SOF

Table 582. OTG_FS input/output signals

Signal name	Signal type	Description
usb_sof	Digital output	USB OTG start-of-frame event for on chip peripherals
usb_wkup	Digital output	USB OTG wakeup event output
usb_gbl_it	Digital output	USB OTG global interrupt

50.4.3 OTG_FS core

The CPU reads and writes from/to the OTG core registers through the AHB peripheral bus. It is informed of USB events through the single USB OTG interrupt line described in *Section 50.13: OTG_FS interrupts*.

The CPU submits data over the USB by writing 32-bit words to dedicated OTG locations (push registers). The data are then automatically stored into Tx-data FIFOs configured within the USB data RAM. There is one Tx FIFO push register for each in-endpoint (peripheral mode) or out-channel (host mode).

The CPU receives the data from the USB by reading 32-bit words from dedicated OTG addresses (pop registers). The data are then automatically retrieved from a shared Rx FIFO configured within the 1.25-Kbyte USB data RAM. There is one Rx FIFO pop register for each out-endpoint or in-channel.

The USB protocol layer is driven by the serial interface engine (SIE) and serialized over the USB by the transceiver module within the on-chip physical layer (PHY).

Caution: To guarantee a correct operation for the USB OTG FS peripheral, the AHB frequency must be higher than 14.2 MHz.

50.4.4 Embedded full-speed OTG PHY connected to OTG_FS

The embedded full-speed OTG PHY is controlled by the OTG FS core and conveys USB control & data signals through the full-speed subset of the UTMI+ Bus (UTMIFS). It provides the physical support to USB connectivity.
The full-speed OTG PHY includes the following components:

- FS/LS transceiver module used by both host and device. It directly drives transmission and reception on the single-ended USB lines.
- DP/DM integrated pull-up and pull-down resistors controlled by the OTG_FS core depending on the current role of the device. As a peripheral, it enables the DP pull-up resistor to signal full-speed peripheral connections as soon as V_{BUS} is sensed to be at a valid level (B-session valid). In host mode, pull-down resistors are enabled on both DP/DM. Pull-up and pull-down resistors are dynamically switched when the role of the device is changed via the host negotiation protocol (HNP).
- Pull-up/pull-down resistor ECN circuit. The DP pull-up consists of two resistors controlled separately from the OTG_FS as per the resistor Engineering Change Notice applied to USB Rev2.0. The dynamic trimming of the DP pull-up strength allows for better noise rejection and Tx/Rx signal quality.

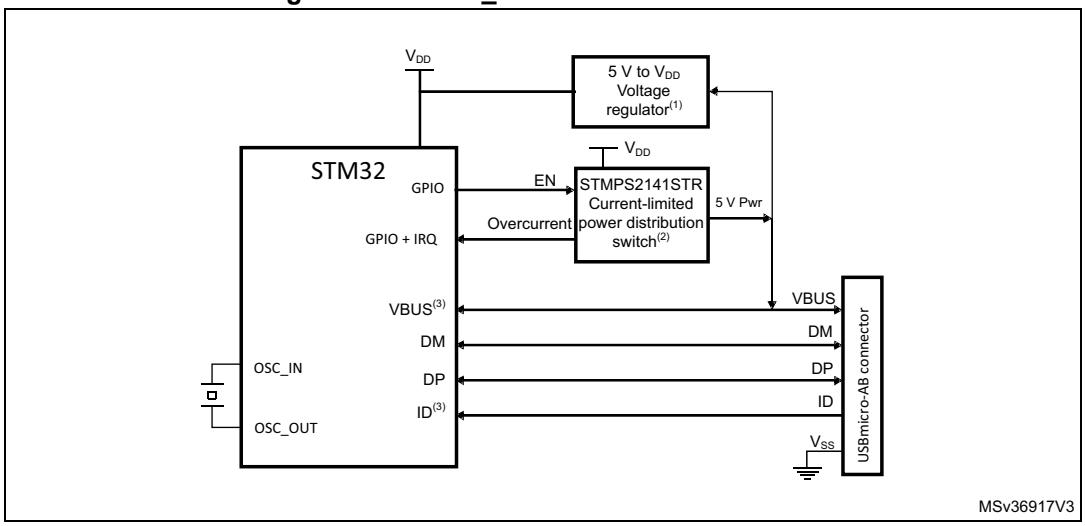
50.4.5 OTG detections

Additionally the OTG_FS uses the following functions:

- integrated ID pull-up resistor used to sample the ID line for A/B device identification.
- V_{BUS} sensing comparators with hysteresis used to detect V_{BUS} valid, A-B session valid and session-end voltage thresholds. They are used to detect valid startup and end-of-session conditions, and constantly monitor the V_{BUS} supply during USB operations.

50.5 OTG_FS dual role device (DRD)

Figure 1011. OTG_FS A-B device connection



1. External voltage regulator only needed when building a VBUS powered device.
2. STMPS2141STR needed only if the application has to support a VBUS powered device. A basic power switch can be used if 5 V are available on the application board.
3. VBUS/ID pin(s) where implemented (see *Table 581: OTG_FS input/output pins*).

50.5.1 ID line detection

The host or peripheral (the default) role is assumed depending on the ID input pin. The ID line status is determined on plugging in the USB cable, depending on whether a MicroA or MicroB plug is connected to the micro-AB receptacle.

- If the B-side of the USB cable is connected with a floating ID wire, the integrated pull-up resistor detects a high ID level and the default peripheral role is confirmed. In this configuration the OTG_FS complies with the standard FSM described in section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.
- If the A-side of the USB cable is connected with a grounded ID, the OTG_FS issues an ID line status change interrupt (CIDSCHG bit in OTG_GINTSTS) for host software initialization, and automatically switches to the host role. In this configuration the OTG_FS complies with the standard FSM described by section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.

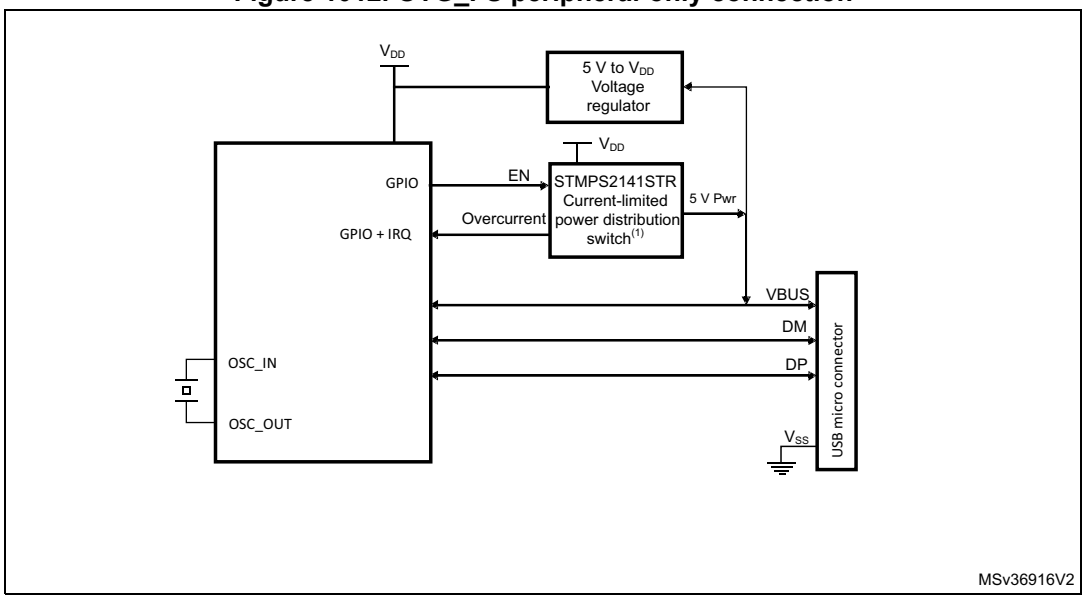
50.6 OTG_FS as a USB peripheral

This section gives the functional description of the OTG_FS in the USB peripheral mode. The OTG_FS works as an USB peripheral in the following circumstances:

- OTG B-Peripheral
  - OTG B-device default state if B-side of USB cable is plugged in
- B-device
  - If the ID line is present, functional and connected to the B-side of the USB cable.
- Peripheral only (see *Figure 1012: OTG_FS peripheral-only connection*)
  - The force device mode bit (FDMOD) in the *Section 50.15.4: OTG USB configuration register (OTG_GUSBCFG)* is set to 1, forcing the OTG_FS core to work as an USB peripheral-only. In this case, the ID line is ignored even if it is present on the USB connector.

Note: To build a bus-powered device implementation in case of the B-device or peripheral-only configuration, an external regulator has to be added, that generates the necessary power-supply from V_{BUS}.

Figure 1012. OTG_FS peripheral-only connection



1. Use a regulator to build a bus-powered device.

50.6.1 Peripheral states

Powered state

The V_{BUS} input detects the B-session valid voltage by which the USB peripheral is allowed to enter the powered state (see USB2.0 section 9.1). The OTG_FS then automatically connects the DP pull-up resistor to signal full-speed device connection to the host and generates the session request interrupt (SRQINT bit in OTG_GINTSTS) to notify the powered state.

The V_{BUS} input also ensures that valid V_{BUS} levels are supplied by the host during USB operations. If a drop in V_{BUS} below B-session valid happens to be detected (for instance because of a power disturbance or if the host port has been switched off), the OTG_FS

automatically disconnects and the session end detected (SEDET bit in OTG_GOTGINT) interrupt is generated to notify that the OTG_FS has exited the powered state.

In the powered state, the OTG_FS expects to receive some reset signaling from the host. No other USB operation is possible. When a reset signaling is received the reset detected interrupt (USBRST in OTG_GINTSTS) is generated. When the reset signaling is complete, the enumeration done interrupt (ENUMDNE bit in OTG_GINTSTS) is generated and the OTG_FS enters the Default state.

Soft disconnect

The powered state can be exited by software with the soft disconnect feature. The DP pull-up resistor is removed by setting the soft disconnect bit in the device control register (SDIS bit in OTG_DCTL), causing a device disconnect detection interrupt on the host side even though the USB cable was not really removed from the host port.

Default state

In the Default state the OTG_FS expects to receive a SET_ADDRESS command from the host. No other USB operation is possible. When a valid SET_ADDRESS command is decoded on the USB, the application writes the corresponding number into the device address field in the device configuration register (DAD bit in OTG_DCFG). The OTG_FS then enters the address state and is ready to answer host transactions at the configured USB address.

Suspended state

The OTG_FS peripheral constantly monitors the USB activity. After counting 3 ms of USB idleness, the early suspend interrupt (ESUSP bit in OTG_GINTSTS) is issued, and confirmed 3 ms later, if appropriate, by the suspend interrupt (USBSUSP bit in OTG_GINTSTS). The device suspend bit is then automatically set in the device status register (SUSPSTS bit in OTG_DSTS) and the OTG_FS enters the suspended state.

The suspended state may optionally be exited by the device itself. In this case the application sets the remote wakeup signaling bit in the device control register (RWUSIG bit in OTG_DCTL) and clears it after 1 to 15 ms.

When a resume signaling is detected from the host, the resume interrupt (WKUPINT bit in OTG_GINTSTS) is generated and the device suspend bit is automatically cleared.

50.6.2 Peripheral endpoints

The OTG_FS core instantiates the following USB endpoints:

- Control endpoint 0:
  - Bidirectional and handles control messages only
  - Separate set of registers to handle in and out transactions
  - Proper control (OTG_DIEPCTL0/OTG_DOEPCTL0), transfer configuration (OTG_DIEPTSIZ0/OTG_DOEPTSIZ0), and status-interrupt

(OTG_DIEPINT0/)OTG_DOEPINT0) registers. The available set of bits inside the control and transfer size registers slightly differs from that of other endpoints

- 5 IN endpoints
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has proper control (OTG_DIEPCTL*x*), transfer configuration (OTG_DIEPTSIZ*x*), and status-interrupt (OTG_DIEPINT*x*) registers
  - The device IN endpoints common interrupt mask register (OTG_DIEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the IN endpoints (EP0 included)
  - Support for incomplete isochronous IN transfer interrupt (IISOIXFR bit in OTG_GINTSTS), asserted when there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).
- 5 OUT endpoints
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has a proper control (OTG_DOEPCTL*x*), transfer configuration (OTG_DOEPTSIZ*x*) and status-interrupt (OTG_DOEPINT*x*) register
  - Device OUT endpoints common interrupt mask register (OTG_DOEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the OUT endpoints (EP0 included)
  - Support for incomplete isochronous OUT transfer interrupt (INCOMPISOOUT bit in OTG_GINTSTS), asserted when there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).

Endpoint control

- The following endpoint controls are available to the application through the device endpoint-*x* IN/OUT control register (OTG_DIEPCTL*x*/OTG_DOEPCTL*x*):
  - Endpoint enable/disable
  - Endpoint activate in current configuration
  - Program USB transfer type (isochronous, bulk, interrupt)
  - Program supported packet size
  - Program Tx FIFO number associated with the IN endpoint
  - Program the expected or transmitted data0/data1 PID (bulk/interrupt only)
  - Program the even/odd frame during which the transaction is received or transmitted (isochronous only)
  - Optionally program the NAK bit to always negative-acknowledge the host regardless of the FIFO status
  - Optionally program the STALL bit to always stall host tokens to that endpoint
  - Optionally program the SNOOP mode for OUT endpoint not to check the CRC field of received data

Endpoint transfer

The device endpoint-*x* transfer size registers (OTG_DIEPTSIZ*x*/OTG_DOEPTSIZ*x*) allow the application to program the transfer size parameters and read the transfer status. Programming must be done before setting the endpoint enable bit in the endpoint control register. Once the endpoint is enabled, these fields are read-only as the OTG_FS core updates them with the current transfer status.

The following transfer parameters can be programmed:

- Transfer size in bytes
- Number of packets that constitute the overall transfer size

Endpoint status/interrupt

The device endpoint-*x* interrupt registers (OTG_DIEPINT*x*/OTG_DOPEPINT*x)* indicate the status of an endpoint with respect to USB- and AHB-related events. The application must read these registers when the OUT endpoint interrupt bit or the IN endpoint interrupt bit in the core interrupt register (OEPINT bit in OTG_GINTSTS or IEPINT bit in OTG_GINTSTS, respectively) is set. Before the application can read these registers, it must first read the device all endpoints interrupt (OTG_DAINT) register to get the exact endpoint number for the device endpoint-*x* interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers

The peripheral core provides the following status checks and interrupt generation:

- Transfer completed interrupt, indicating that data transfer was completed on both the application (AHB) and USB sides
- Setup stage has been done (control-out only)
- Associated transmit FIFO is half or completely empty (in endpoints)
- NAK acknowledge has been transmitted to the host (isochronous-in only)
- IN token received when Tx FIFO was empty (bulk-in/interrupt-in only)
- Out token received when endpoint was not yet enabled
- Babble error condition has been detected
- Endpoint disable by application is effective
- Endpoint NAK by application is effective (isochronous-in only)
- More than 3 back-to-back setup packets were received (control-out only)
- Timeout condition detected (control-in only)
- Isochronous out packet has been dropped, without generating an interrupt

50.7 OTG_FS as a USB host

This section gives the functional description of the OTG_FS in the USB host mode. The OTG_FS works as a USB host in the following circumstances:

Automatic host mode direct from ID pin:

The ID pin is not always available, refer to product datasheet for available pins.

- Use case:
  - Micro connector: A-Device with a Micro-A plug inserted, and connector ID pin connected to the STM32 ID pin which is then detected in a low state.
- Note that the integrated pull-down resistors are automatically set on the DP/DM lines.

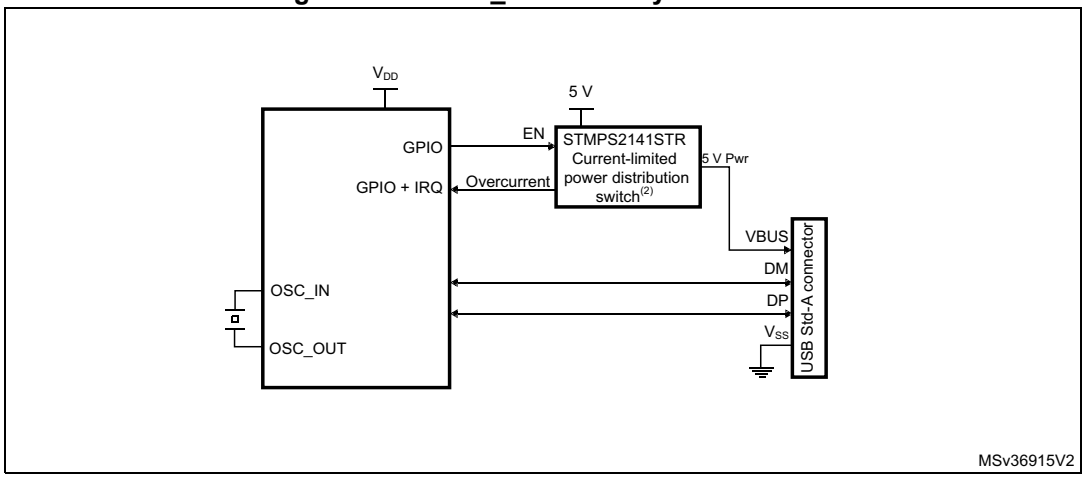
Manual forcing of host mode when not possible via ID pin:

The force host mode bit (FHMOD) in the *OTG USB configuration register (OTG_GUSBCFG)* forces the OTG_FS core to work as a USB host-only when required.

- Use cases:
  - Standard-A connector which always implies Host mode.
  - Micro connector, so ID present on connector, but ID pin not available in pinout, at A-plug insertion (detected by another means, for example: GPIO).
  - A Type-C connector, when host functionality is determined at cable attachment or via power delivery messages.
- Note that the integrated pulldown resistors are automatically set on the DP/DM lines

Note: On-chip 5 V V_{BUS} generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch must be added externally to drive the 5 V V_{BUS} line. The external charge pump can be driven by any GPIO output. This is required for the OTG A-host, A-device and host-only configurations.

Figure 1013. OTG_FS host-only connection



1. V_{DD} range is between 2 V and 3.6 V.

50.7.1 USB host states

Host port power

On-chip 5 V V_{BUS} generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch, must be added externally to drive the 5 V V_{BUS} line. The external charge pump can be driven by any GPIO output or via an I^2C interface connected to an external PMIC (power management IC). When the application decides to power on V_{BUS}, it must also set the port power bit in the host port control and status register (PPWR bit in OTG_HPRT).

V_{BUS} valid

In Host mode, the VBUS sensing pin does not need to be connected to V_{BUS}.

The charge pump overcurrent flag can also be used to prevent electrical damage. Connect the overcurrent flag output from the charge pump to any GPIO input and configure it to

generate a port interrupt on the active level. The overcurrent ISR must promptly disable the V_{BUS} generation and clear the port power bit.

Host detection of a peripheral connection

USB peripherals or B-device are detected as soon as they are connected. The OTG_FS core issues a host port interrupt triggered by the device connected bit in the host port control and status (PCDET bit in OTG_HPRT).

Host detection of peripheral a disconnection

The peripheral disconnection event triggers the disconnect detected interrupt (DISCINT bit in OTG_GINTSTS).

Host enumeration

After detecting a peripheral connection the host must start the enumeration process by sending USB reset and configuration commands to the new peripheral.

The application drives a USB reset signaling (single-ended zero) over the USB by keeping the port reset bit set in the host port control and status register (PRST bit in OTG_HPRT) for a minimum of 10 ms and a maximum of 20 ms. The application takes care of the timing count and then of clearing the port reset bit.

Once the USB reset sequence has completed, the host port interrupt is triggered by the port enable/disable change bit (PENCHNG bit in OTG_HPRT). This informs the application that the speed of the enumerated peripheral can be read from the port speed field in the host port control and status register (PSPD bit in OTG_HPRT) and that the host is starting to drive SOFs (FS) or Keep alives (LS). The host is now ready to complete the peripheral enumeration by sending peripheral configuration commands.

Host suspend

The application decides to suspend the USB activity by setting the port suspend bit in the host port control and status register (PSUSP bit in OTG_HPRT). The OTG_FS core stops sending SOFs and enters the suspended state.

The suspended state can be optionally exited on the remote device's initiative (remote wakeup). In this case the remote wakeup interrupt (WKUPINT bit in OTG_GINTSTS) is generated upon detection of a remote wakeup signaling, the port resume bit in the host port control and status register (PRES bit in OTG_HPRT) self-sets, and resume signaling is automatically driven over the USB. The application must time the resume window and then clear the port resume bit to exit the suspended state and restart the SOF.

If the suspended state is exited on the host initiative, the application must set the port resume bit to start resume signaling on the host port, time the resume window and finally clear the port resume bit.

50.7.2 Host channels

The OTG_FS core instantiates 12 host channels. Each host channel supports an USB host transfer (USB pipe). The host is not able to support more than 12 transfer requests at the same time. If more than 12 transfer requests are pending from the application, the host controller driver (HCD) must re-allocate channels when they become available from previous duty, that is, after receiving the transfer completed and channel halted interrupts.

Each host channel can be configured to support in/out and any type of periodic/nonperiodic transaction. Each host channel makes us of proper control (OTG_HCCHAR*x*), transfer configuration (OTG_HCTSIZ*x*) and status/interrupt (OTG_HCINT*x*) registers with associated mask (OTG_HCINTMSK*x*) registers.

Host channel control

- The following host channel controls are available to the application through the host channel-*x* characteristics register (OTG_HCCHAR*x*):
  - Channel enable/disable
  - Program the FS/LS speed of target USB peripheral
  - Program the address of target USB peripheral
  - Program the endpoint number of target USB peripheral
  - Program the transfer IN/OUT direction
  - Program the USB transfer type (control, bulk, interrupt, isochronous)
  - Program the maximum packet size (MPS)
  - Program the periodic transfer to be executed during odd/even frames

Host channel transfer

The host channel transfer size registers (OTG_HCTSIZ*x*) allow the application to program the transfer size parameters, and read the transfer status. Programming must be done before setting the channel enable bit in the host channel characteristics register. Once the endpoint is enabled the packet count field is read-only as the OTG_FS core updates it according to the current transfer status.

- The following transfer parameters can be programmed:
  - transfer size in bytes
  - number of packets making up the overall transfer size
  - initial data PID

Host channel status/interrupt

The host channel-*x* interrupt register (OTG_HCINT*x*) indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read these register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_GINTSTS) is set. Before the application can read these registers, it must first read the host all channels interrupt (OTG_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers.

The mask bits for each interrupt source of each channel are also available in the OTG_HCINTMSKx register.

- The host core provides the following status checks and interrupt generation:
  - Transfer completed interrupt, indicating that the data transfer is complete on both the application (AHB) and USB sides
  - Channel has stopped due to transfer completed, USB transaction error or disable command from the application
  - Associated transmit FIFO is half or completely empty (IN endpoints)
  - ACK response received
  - NAK response received
  - STALL response received
  - USB transaction error due to CRC failure, timeout, bit stuff error, false EOP
  - Babble error
  - frame overrun
  - data toggle error

50.7.3 Host scheduler

The host core features a built-in hardware scheduler which is able to autonomously re-order and manage the USB transaction requests posted by the application. At the beginning of each frame the host executes the periodic (isochronous and interrupt) transactions first, followed by the nonperiodic (control and bulk) transactions to achieve the higher level of priority granted to the isochronous and interrupt transfer types by the USB specification.

The host processes the USB transactions through request queues (one for periodic and one for nonperiodic). Each request queue can hold up to 8 entries. Each entry represents a pending transaction request from the application, and holds the IN or OUT channel number along with other information to perform a transaction on the USB. The order in which the requests are written to the queue determines the sequence of the transactions on the USB interface.

At the beginning of each frame, the host processes the periodic request queue first, followed by the nonperiodic request queue. The host issues an incomplete periodic transfer interrupt (IPXFR bit in OTG_GINTSTS) if an isochronous or interrupt transaction scheduled for the current frame is still pending at the end of the current frame. The OTG_FS core is fully responsible for the management of the periodic and nonperiodic request queues.The periodic transmit FIFO and queue status register (OTG_HPTXSTS) and nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) are read-only registers which can be used by the application to read the status of each request queue. They contain:

- The number of free entries currently available in the periodic (nonperiodic) request queue (8 max)
- Free space currently available in the periodic (nonperiodic) Tx FIFO (out-transactions)
- IN/OUT token, host channel number and other status information.

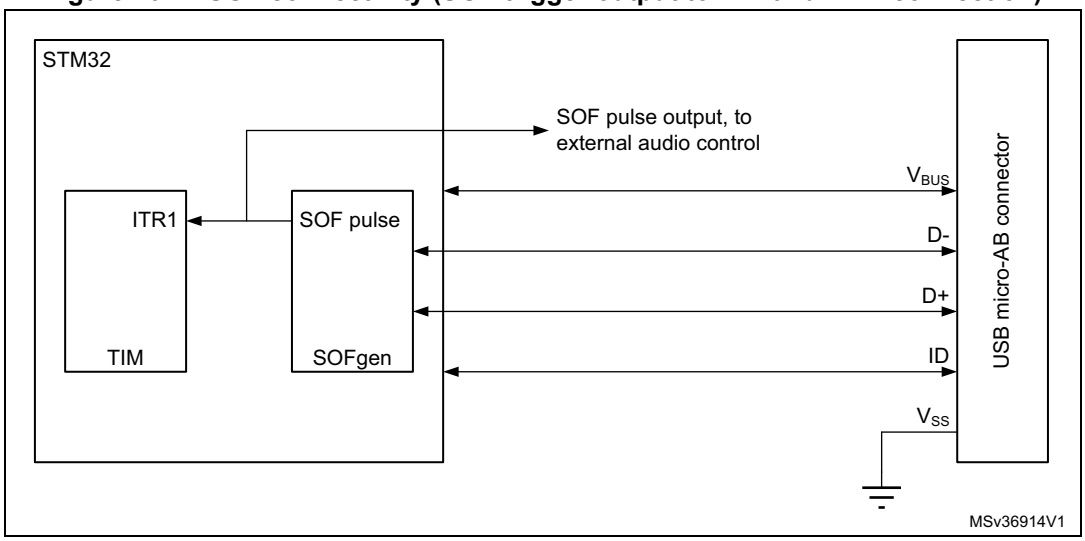
As request queues can hold a maximum of 8 entries each, the application can push to schedule host transactions in advance with respect to the moment they physically reach the SB for a maximum of 8 pending periodic transactions plus 8 pending non-periodic transactions.

To post a transaction request to the host scheduler (queue) the application must check that there is at least 1 entry available in the periodic (nonperiodic) request queue by reading the

PTXQSAV bits in the OTG_HNPTXSTS register or NPTQXSAV bits in the OTG_HNPTXSTS register.

50.8 OTG_FS SOF trigger

Figure 1014. SOF connectivity (SOF trigger output to TIM and ITR1 connection)



The OTG_FS core provides means to monitor, track and configure SOF framing in the host and peripheral, as well as an SOF pulse output connectivity feature.

Such utilities are especially useful for adaptive audio clock generation techniques, where the audio peripheral needs to synchronize to the isochronous stream provided by the PC, or the host needs to trim its framing rate according to the requirements of the audio peripheral.

50.8.1 Host SOFs

In host mode the number of PHY clocks occurring between the generation of two consecutive SOF (FS) or Keep-alive (LS) tokens is programmable in the host frame interval register (HFIR), thus providing application control over the SOF framing period. An interrupt is generated at any start of frame (SOF bit in OTG_GINTSTS). The current frame number and the time remaining until the next SOF are tracked in the host frame number register (HFNUM).

A SOF pulse signal, is generated at any SOF starting token and with a width of 20 HCLK cycles. The SOF pulse is also internally connected to the input trigger of the timer, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.

50.8.2 Peripheral SOFs

In device mode, the start of frame interrupt is generated each time an SOF token is received on the USB (SOF bit in OTG_GINTSTS). The corresponding frame number can be read from the device status register (FNSOF bit in OTG_DSTS). A SOF pulse signal with a width of 20 HCLK cycles is also generated.The SOF pulse signal is also internally connected to the TIM input trigger, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.

The end of periodic frame interrupt (OTG_GINTSTS/EOPF) is used to notify the application when 80%, 85%, 90% or 95% of the time frame interval elapsed depending on the periodic frame interval field in the device configuration register (PFIVL bit in OTG_DCFG). This feature can be used to determine if all of the isochronous traffic for that frame is complete.

50.9 OTG_FS low-power modes

Table 583 below defines the STM32 low power modes and their compatibility with the OTG.

Table 583. Compatibility of STM32 low power modes with the OTG

Mode	Description	USB compatibility
Run	MCU fully active	Required when USB not in suspend state.
Sleep	USB suspend exit causes the device to exit Sleep mode. Peripheral registers content is kept.	Available while USB is in suspend state.
Stop	USB suspend exit causes the device to exit Stop mode. Peripheral registers content is kept(1).	Available while USB is in suspend state.
Standby	Powered-down. The peripheral must be reinitialized after exiting Standby mode.	Not compatible with USB applications.

1. Within Stop mode there are different possible settings. Some restrictions may also exist, refer to *Section 6: Power control (PWR)* to understand which (if any) restrictions apply when using OTG.

The following bits and procedures reduce power consumption.

The power consumption of the OTG PHY is controlled by the following bit(s) in the general core configuration register, depending on OTG revision supported.

- PHY power down (OTG_GCCFG/PWRDWN)

  It switches on/off the full-speed transceiver module of the PHY. It must be preliminarily set to allow any USB operation
- V_{BUS} detection enable (OTG_GCCFG/VBDEN)

  It switches on/off the V_{BUS} sensing comparators associated with OTG operations

Power reduction techniques are available while in the USB suspended state, when the USB session is not yet valid or the device is disconnected.

- Stop PHY clock (STPPCLK bit in OTG_PCGCCTL)

  When setting the stop PHY clock bit in the clock gating control register, most of the 48 MHz clock domain internal to the OTG core is switched off by clock gating. The dynamic power consumption due to the USB clock switching activity is cut even if the 48 MHz clock input is kept running by the application

  Most of the transceiver is also disabled, and only the part in charge of detecting the asynchronous resume or remote wakeup event is kept alive.
- Gate HCLK (GATEHCLK bit in OTG_PCGCCTL)

  When setting the Gate HCLK bit in the clock gating control register, most of the system clock domain internal to the OTG_FS core is switched off by clock gating. Only the register read and write interface is kept alive. The dynamic power consumption due to

the USB clock switching activity is cut even if the system clock is kept running by the application for other purposes.

- USB system stop

  When the OTG_FS is in the USB suspended state, the application may decide to drastically reduce the overall power consumption by a complete shut down of all the clock sources in the system. USB System Stop is activated by first setting the Stop PHY clock bit and then configuring the system deep sleep mode in the power control system module (PWR).

  The OTG_FS core automatically reactivates both system and USB clocks by asynchronous detection of remote wakeup (as an host) or resume (as a device) signaling on the USB.

To save dynamic power, the USB data FIFO is clocked only when accessed by the OTG_FS core.

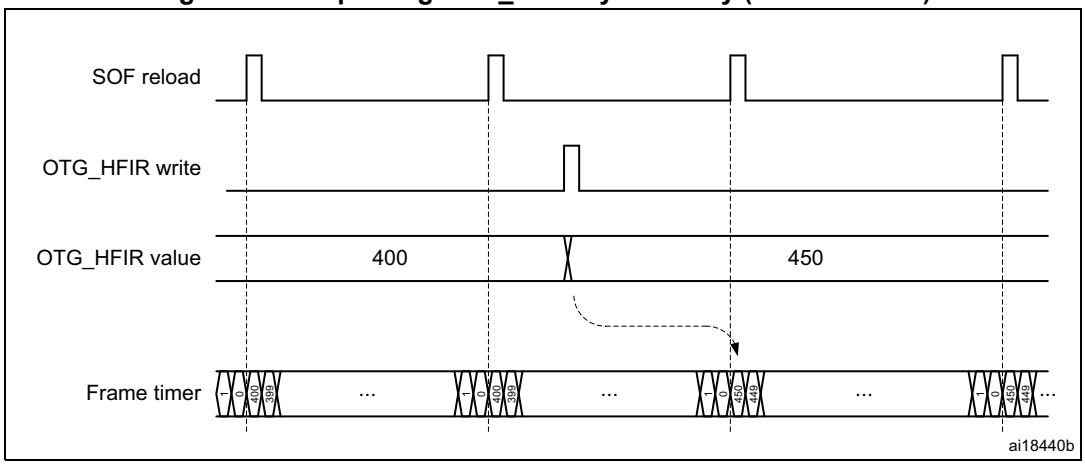
50.10 OTG_FS Dynamic update of the OTG_HFIR register

The USB core embeds a dynamic trimming capability of SOF framing period in host mode allowing to synchronize an external device with the SOF frames.

When the OTG_HFIR register is changed within a current SOF frame, the SOF period correction is applied in the next frame as described in *Figure 1015*.

For a dynamic update, it is required to set RLDCTRL=1.

Figure 1015. Updating OTG_HFIR dynamically (RLDCTRL = 1)

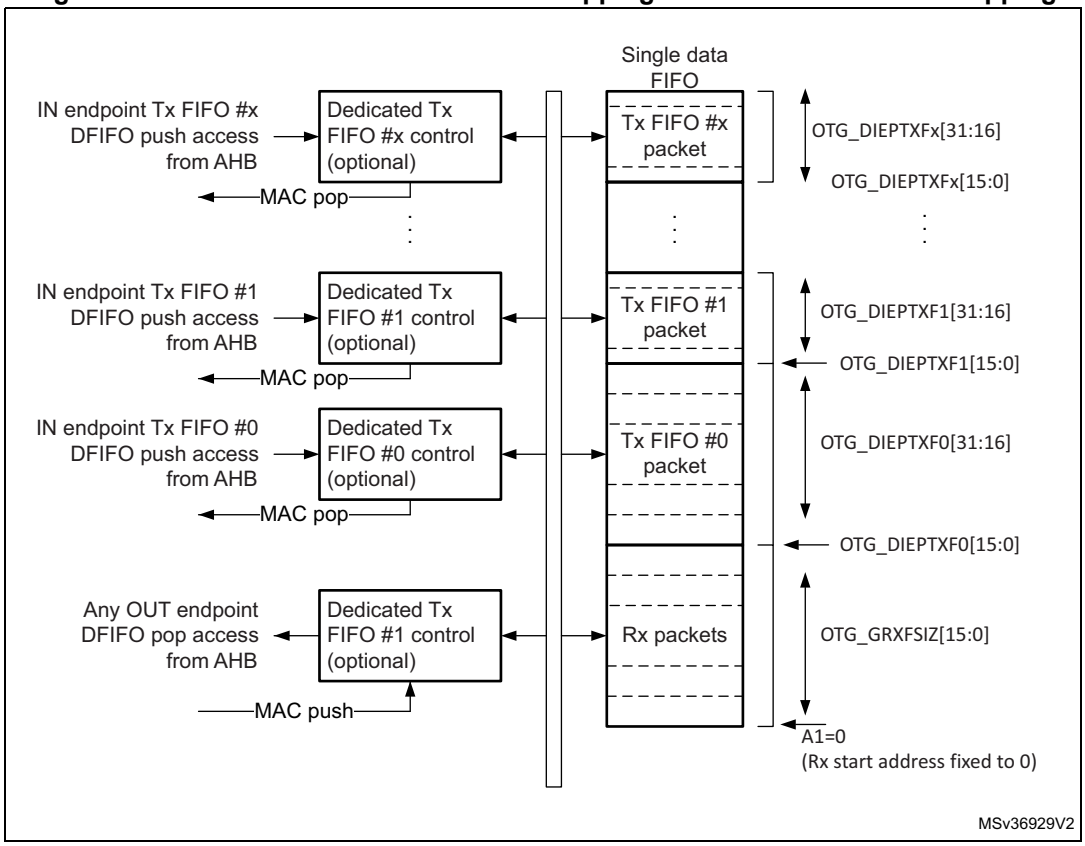


50.11 OTG_FS data FIFOs

The USB system features 1.25 Kbytes of dedicated RAM with a sophisticated FIFO control mechanism. The packet FIFO controller module in the OTG_FS core organizes RAM space into Tx FIFOs into which the application pushes the data to be temporarily stored before the USB transmission, and into a single Rx FIFO where the data received from the USB are temporarily stored before retrieval (popped) by the application. The number of FIFOs to be created and how these are organized inside the RAM depends on the device's role. In peripheral mode an additional Tx FIFO must be created for each active IN endpoint. Any FIFO size is software configured to better meet the application requirements.

50.11.1 Peripheral FIFO architecture

Figure 1016. Device-mode FIFO address mapping and AHB FIFO access mapping



Peripheral Rx FIFO

The OTG peripheral uses a single receive FIFO that receives the data directed to all OUT endpoints. Received packets are stacked back-to-back until free space is available in the Rx FIFO. The status of the received packet (which contains the OUT endpoint destination number, the byte count, the data PID and the validity of the received data) is also stored by the core on top of the data payload. When no more space is available, host transactions are NACKed and an interrupt is received on the addressed endpoint. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it more efficient for the USB peripheral to fill in the receive RAM buffer:

- All OUT endpoints share the same RAM buffer (shared FIFO)
- The OTG_FS core can fill in the receive FIFO up to the limit for any host sequence of OUT tokens

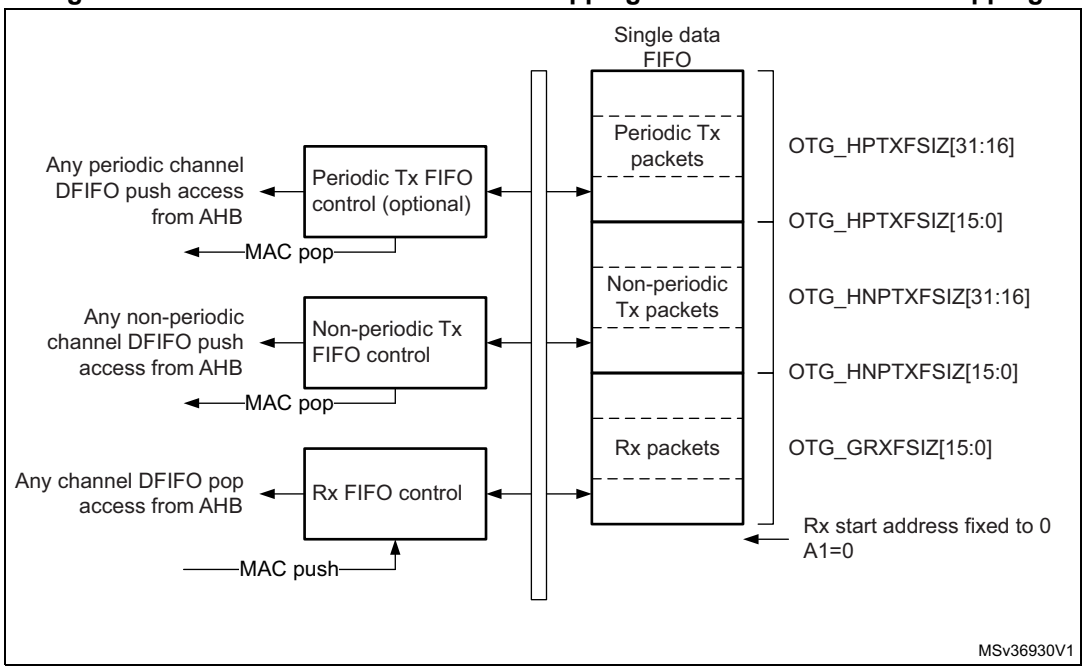
The application keeps receiving the Rx FIFO non-empty interrupt (RXFLVL bit in OTG_GINTSTS) as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register (OTG_GRXSTSP) and finally pops data off the receive FIFO by reading from the endpoint-related pop address.

Peripheral Tx FIFOs

The core has a dedicated FIFO for each IN endpoint. The application configures FIFO sizes by writing the endpoint 0 transmit FIFO size register (OTG_DIEPTXF0) for IN endpoint0 and the device IN endpoint transmit FIFOx registers (OTG_DIEPTXFx) for IN endpoint-x.

50.11.2 Host FIFO architecture

Figure 1017. Host-mode FIFO address mapping and AHB FIFO access mapping



Host Rx FIFO

The host uses one receiver FIFO for all periodic and nonperiodic transactions. The FIFO is used as a receive buffer to hold the received data (payload of the received packet) from the USB until it is transferred to the system memory. Packets received from any remote IN endpoint are stacked back-to-back until free space is available. The status of each received packet with the host channel destination, byte count, data PID and validity of the received data are also stored into the FIFO. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it highly efficient for the USB host to fill in the receive data buffer:

- All IN configured host channels share the same RAM buffer (shared FIFO)
- The OTG_FS core can fill in the receive FIFO up to the limit for any sequence of IN tokens driven by the host software

The application receives the Rx FIFO not-empty interrupt as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register and finally pops the data off the receive FIFO.

Host Tx FIFOs

The host uses one transmit FIFO for all non-periodic (control and bulk) OUT transactions and one transmit FIFO for all periodic (isochronous and interrupt) OUT transactions. FIFOs are used as transmit buffers to hold the data (payload of the transmit packet) to be transmitted over the USB. The size of the periodic (nonperiodic) Tx FIFO is configured in the host periodic (nonperiodic) transmit FIFO size OTG_HPTXFSIZ / OTG_HNPTXFSIZ) register.

The two Tx FIFO implementation derives from the higher priority granted to the periodic type of traffic over the USB frame. At the beginning of each frame, the built-in host scheduler processes the periodic request queue first, followed by the nonperiodic request queue.

The two transmit FIFO architecture provides the USB host with separate optimization for periodic and nonperiodic transmit data buffer management:

- All host channels configured to support periodic (nonperiodic) transactions in the OUT direction share the same RAM buffer (shared FIFOs)
- The OTG_FS core can fill in the periodic (nonperiodic) transmit FIFO up to the limit for any sequence of OUT tokens driven by the host software

The OTG_FS core issues the periodic Tx FIFO empty interrupt (PTXFE bit in OTG_GINTSTS) as long as the periodic Tx FIFO is half or completely empty, depending on the value of the periodic Tx FIFO empty level bit in the AHB configuration register (PTXFELVL bit in OTG_GAHBCFG). The application can push the transmission data in advance as long as free space is available in both the periodic Tx FIFO and the periodic request queue. The host periodic transmit FIFO and queue status register (OTG_HPTXSTS) can be read to know how much space is available in both.

OTG_FS core issues the non periodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) as long as the nonperiodic Tx FIFO is half or completely empty depending on the non periodic Tx FIFO empty level bit in the AHB configuration register (TXFELVL bit in OTG_GAHBCFG). The application can push the transmission data as long as free space is available in both the nonperiodic Tx FIFO and nonperiodic request queue. The host nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) can be read to know how much space is available in both.

50.11.3 FIFO RAM allocation

Device mode

Receive FIFO RAM allocation: the application must allocate RAM for SETUP packets:

- 13 locations must be reserved in the receive FIFO to receive SETUP packets on control endpoint. The core does not use these locations, which are reserved for SETUP packets, to write any other data.
- One location is to be allocated for Global OUT NAK.
- Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size / 4) + 1 must be allocated to receive packets. If multiple isochronous endpoints or a single high-bandwidth isochronous endpoint is enabled, at least two ((largest packet size / 4) + 1) spaces must be allocated to receive back-to-back packets. Typically, two ((largest packet size / 4) + 1) spaces are

recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.

- Along with the last packet for each endpoint, transfer complete status information is also pushed to the FIFO. One location for each OUT endpoint is recommended.

Device RxFIFO =

(5 + 8) + ((largest USB packet used / 4) + 1 for status information) + (2 × number of OUT endpoints) + 1 for Global NAK

Example: The MPS is 1,023 bytes for a periodic USB packet and 64 bytes for a nonperiodic USB packet. There are six OUT endpoints, six IN endpoints, including one control endpoint.

Example (Full Speed): Device RxFIFO =

(5 + 8) + ((1023 / 4) +1) + (2 × 6) + 1 = 283

Transmit FIFO RAM allocation: the minimum RAM space required for each IN endpoint Transmit FIFO is the maximum packet size for that particular IN endpoint.

Note: More space allocated in the transmit IN endpoint FIFO results in better performance on the USB.

Host mode

Receive FIFO RAM allocation:

Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size / 4) + 1 must be allocated to receive packets. If either multiple isochronous endpoints, or a single high bandwidth isochronous endpoint is enabled, then at least two (largest packet size / 4) + 1 spaces must be allocated to receive back-to-back packets. Typically, two (largest packet size / 4) + 1 spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.

Along with the last packet in the host channel, transfer complete status information is also pushed to the FIFO. So one location must be allocated for this.

Host RxFIFO =

(largest USB packet used / 4) + 1 for status information + 1 transfer Complete

Example (Full Speed): Host RxFIFO =

((64 / 4) +1) + 1 = 18

Transmit FIFO RAM allocation:

The minimum amount of RAM required for the host Non-periodic Transmit FIFO is the largest maximum packet size among all supported non-periodic OUT channels.

Typically, two largest packet sizes worth of space is recommended, so that when the current packet is under transfer to the USB, the CPU can get the next packet.

Non-Periodic TxFIFO = largest non-periodic USB packet used / 4

Example: The MPS is 64 bytes for a nonperiodic USB packet.

Nonperiodic TxFIFO = (64 / 4) = 16

The minimum amount of RAM required for host periodic Transmit FIFO is the largest maximum packet size out of all the supported periodic OUT channels. If there is at least one isochronous OUT endpoint, then the space must be at least two times the maximum packet size of that channel.

Host Periodic TxFIFO =

Sum total of (largest periodic USB packet used × number of packets per micro-frame) ÷ 4 for all periodic channels.

Example: The MPS is 1,023 bytes for a periodic USB packet. One endpoint is active and a single packet is transmitted in each frame:

Host Periodic TxFIFO = ((1 × 1,023) / 4) = 256

Note: More space allocated in the Transmit Non-periodic FIFO results in better performance on the USB.

50.12 OTG_FS system performance

Best USB and system performance is achieved owing to the large RAM buffers, the highly configurable FIFO sizes, the quick 32-bit FIFO access through AHB push/pop registers and, especially, the advanced FIFO control mechanism. Indeed, this mechanism allows the OTG_FS to fill in the available RAM space at best regardless of the current USB sequence. With these features:

- The application gains good margins to calibrate its intervention in order to optimize the CPU bandwidth usage:
  - It can accumulate large amounts of transmission data in advance compared to when they are effectively sent over the USB
  - It benefits of a large time margin to download data from the single receive FIFO
- The USB core is able to maintain its full operating rate, that is to provide maximum full-speed bandwidth with a great margin of autonomy versus application intervention:
  - It has a large reserve of transmission data at its disposal to autonomously manage the sending of data over the USB
  - It has a lot of empty space available in the receive buffer to autonomously fill it in with the data coming from the USB

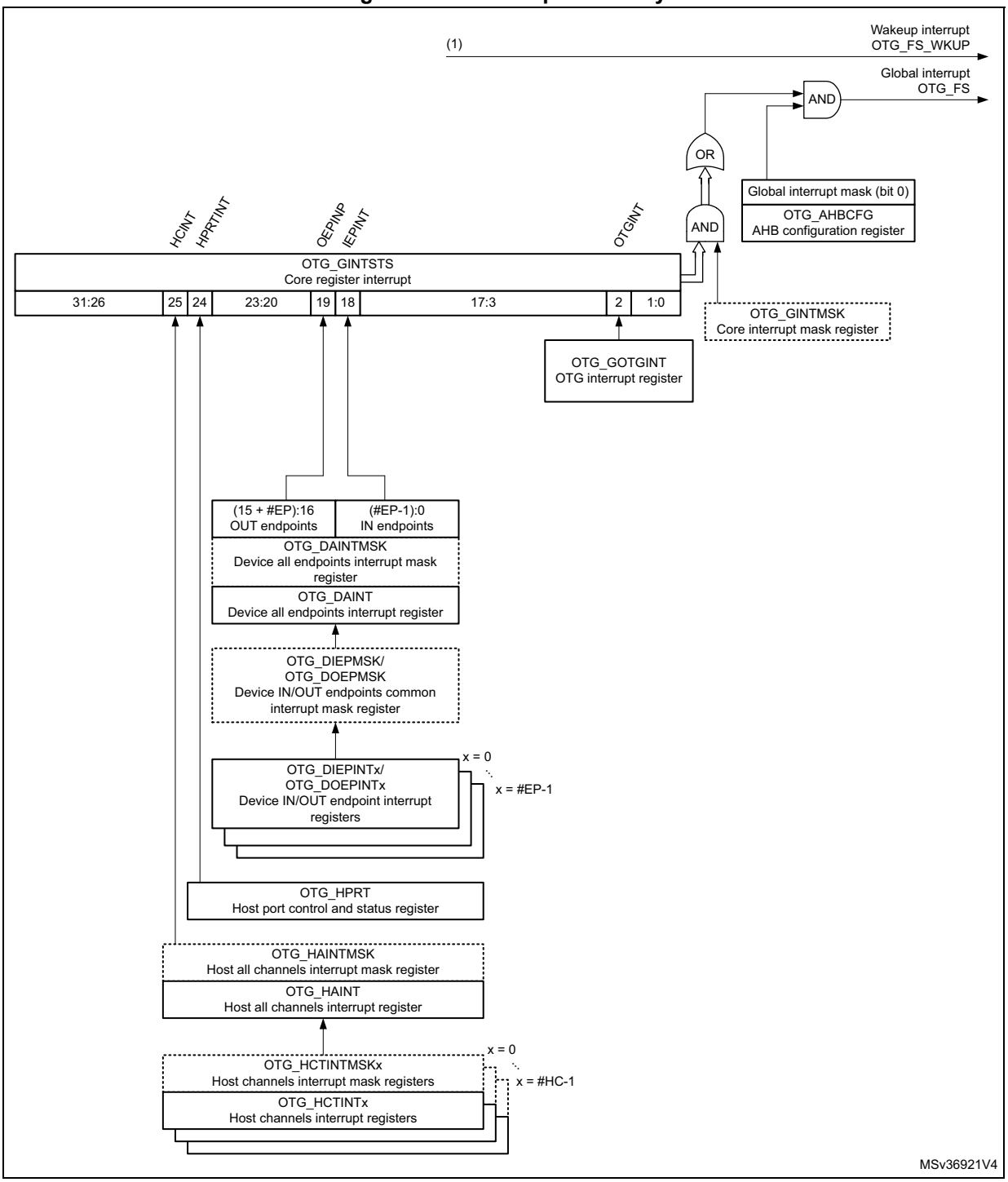
As the OTG_FS core is able to fill in the 1.25-Kbyte RAM buffer very efficiently, and as 1.25-Kbyte of transmit/receive data is more than enough to cover a full speed frame, the USB system is able to withstand the maximum full-speed data rate for up to one USB frame (1 ms) without any CPU intervention.

50.13 OTG_FS interrupts

When the OTG_FS controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

Figure 1018 shows the interrupt hierarchy.

Figure 1018. Interrupt hierarchy



1. OTG_FS_WKUP becomes active (high state) when resume condition occurs during L1 SLEEP or L2 SUSPEND states.

50.14 OTG_FS control and status registers

By reading from and writing to the control and status registers (CSRs) through the AHB slave interface, the application controls the OTG_FS controller. These registers are 32 bits wide, and the addresses are 32-bit block aligned. The OTG_FS registers must be accessed by words (32 bits).

CSRs are classified as follows:

- Core global registers
- Host-mode registers
- Host global registers
- Host port CSRs
- Host channel-specific registers
- Device-mode registers
- Device global registers
- Device endpoint-specific registers
- Power and clock-gating registers
- Data FIFO (DFIFO) access registers

Only the core global, power and clock-gating, data FIFO access, and host port control and status registers can be accessed in both host and device modes. When the OTG_FS controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

50.14.1 CSR memory map

The host and device mode registers occupy different addresses. All registers are implemented in the AHB clock domain.

Global CSR map

These registers are available in both host and device modes.

Table 584. Core global control and status registers (CSRs)

Acronym	Address offset	Register name
OTG_GOTGCTL	0x000	*Section 50.15.1: OTG control and status register (OTG_GOTGCTL)*
OTG_GOTGINT	0x004	*Section 50.15.2: OTG interrupt register (OTG_GOTGINT)*
OTG_GAHBCFG	0x008	*Section 50.15.3: OTG AHB configuration register (OTG_GAHBCFG)*
OTG_GUSBCFG	0x00C	*Section 50.15.4: OTG USB configuration register (OTG_GUSBCFG)*
OTG_GRSTCTL	0x010	*Section 50.15.5: OTG reset register (OTG_GRSTCTL)*
OTG_GINTSTS	0x014	*Section 50.15.6: OTG core interrupt register [alternate] (OTG_GINTSTS)* *Section 50.15.7: OTG core interrupt register [alternate] (OTG_GINTSTS)*

Table 584. Core global control and status registers (CSRs) (continued)

Acronym	Address offset	Register name
OTG_GINTMSK	0x018	*Section 50.15.8: OTG interrupt mask register [alternate] (OTG_GINTMSK)* *Section 50.15.9: OTG interrupt mask register [alternate] (OTG_GINTMSK)*
OTG_GRXSTSR	0x01C	*Section 50.15.10: OTG receive status debug read [alternate] (OTG_GRXSTSR)* *Section 50.15.11: OTG receive status debug read register [alternate] (OTG_GRXSTSR)*
OTG_GRXSTSP	0x020	*Section 50.15.12: OTG status read and pop registers [alternate] (OTG_GRXSTSP)* *Section 50.15.13: OTG status read and pop registers [alternate] (OTG_GRXSTSP)*
OTG_GRXFSIZ	0x024	*Section 50.15.14: OTG receive FIFO size register (OTG_GRXFSIZ)*
OTG_HNPTXFSIZ/ OTG_DIEPTXF0(1)	0x028	*Section 50.15.15: OTG host non-periodic transmit FIFO size register [alternate] (OTG_HNPTXFSIZ)* *Section 50.15.16: Endpoint 0 Transmit FIFO size [alternate] (OTG_DIEPTXF0)*
OTG_HNPTXSTS	0x02C	*Section 50.15.17: OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)*
OTG_GCCFG	0x038	*Section 50.15.18: OTG general core configuration register (OTG_GCCFG)*
OTG_CID	0x03C	*Section 50.15.19: OTG core ID register (OTG_CID)*
OTG_GLPMCFG	0x54	*Section 50.15.20: OTG core LPM configuration register (OTG_GLPMCFG)*
OTG_HPTXFSIZ	0x100	*Section 50.15.21: OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)*
OTG_DIEPTXFx	0x104 0x108 ... 0x114	*Section 50.15.22: OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx)*

1. The general rule is to use OTG_HNPTXFSIZ for host mode and OTG_DIEPTXF0 for device mode.

Host-mode CSR map

These registers must be programmed every time the core changes to host mode.

Table 585. Host-mode control and status registers (CSRs)

Acronym	Offset address	Register name
OTG_HCFG	0x400	*Section 50.15.24: OTG host configuration register (OTG_HCFG)*
OTG_HFIR	0x404	*Section 50.15.25: OTG host frame interval register (OTG_HFIR)*
OTG_HFNUM	0x408	*Section 50.15.26: OTG host frame number/frame time remaining register (OTG_HFNUM)*

Table 585. Host-mode control and status registers (CSRs) (continued)

Acronym	Offset address	Register name
OTG_HPTXSTS	0x410	*Section 50.15.27: OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)*
OTG_HAINT	0x414	*Section 50.15.28: OTG host all channels interrupt register (OTG_HAINT)*
OTG_HAINTMSK	0x418	*Section 50.15.29: OTG host all channels interrupt mask register (OTG_HAINTMSK)*
OTG_HPRT	0x440	*Section 50.15.30: OTG host port control and status register (OTG_HPRT)*
OTG_HCCHARx	0x500 0x520 ... 0x660	*Section 50.15.31: OTG host channel x characteristics register (OTG_HCCHARx)*
OTG_HCINTx	0x508 0x528 .... 0x668	*Section 50.15.32: OTG host channel x interrupt register (OTG_HCINTx)*
OTG_HCINTMSKx	0x50C 0x52C 0x66C	*Section 50.15.33: OTG host channel x interrupt mask register (OTG_HCINTMSKx)*
OTG_HCTSIZx	0x510 0x530 .... 0x670	*Section 50.15.34: OTG host channel x transfer size register (OTG_HCTSIZx)*

Device-mode CSR map

These registers must be programmed every time the core changes to device mode.

Table 586. Device-mode control and status registers

Acronym	Offset address	Register name
OTG_DCFG	0x800	*Section 50.15.36: OTG device configuration register (OTG_DCFG)*
OTG_DCTL	0x804	*Section 50.15.37: OTG device control register (OTG_DCTL)*
OTG_DSTS	0x808	*Section 50.15.38: OTG device status register (OTG_DSTS)*
OTG_DIEPMSK	0x810	*Section 50.15.39: OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)*
OTG_DOEPMSK	0x814	*Section 50.15.40: OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK)*
OTG_DAINT	0x818	*Section 50.15.41: OTG device all endpoints interrupt register (OTG_DAINT)*

Table 586. Device-mode control and status registers (continued)

Acronym	Offset address	Register name
OTG_DAINTMSK	0x81C	*Section 50.15.42: OTG all endpoints interrupt mask register (OTG_DAINTMSK)*
OTG_DIEPEMPMSK	0x834	*Section 50.15.43: OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK)*
OTG_DEACHINT	0x838	*Section 50.15.44: OTG device each endpoint interrupt register (OTG_DEACHINT)*
OTG_DEACHINTMSK	0x83C	*Section 50.15.45: OTG device each endpoint interrupt mask register (OTG_DEACHINTMSK)*
OTG_DIEPEACHM	0x844	*Section 50.15.46: OTG device each IN endpoint-1 interrupt mask register (OTG_DIEPEACHM)*
OTG_DOEPEACHMSK1	0x884	*Section 50.15.47: OTG device each OUT endpoint-1 interrupt mask register (OTG_DOEPEACHMSK1)*
OTG_DIEPCTL0	0x900	*Section 50.15.48: OTG device control IN endpoint 0 control register (OTG_DIEPCTL0)*
OTG_DIEPCTLx	0x920 0x940 ... 0x9A0	*Section 50.15.49: OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx)* *Section 50.15.50: OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx)*
OTG_DIEPINTx	0x908 0x928 .... 0x988	*Section 50.15.51: OTG device IN endpoint x interrupt register (OTG_DIEPINTx)*
OTG_DIEPTSIZ0	0x910	*Section 50.15.52: OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0)*
OTG_DTXFSTSx	0x918 0x938 .... 0x998	*Section 50.15.53: OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx)*
OTG_DIEPTSIZx	0x930 0x950 ... 0x9B0	*Section 50.15.54: OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)*
OTG_DOEPCTL0	0xB00	*Section 50.15.55: OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)*
OTG_DOEPINTx	0xB08 0xB28 ... 0xBA8	*Section 50.15.56: OTG device OUT endpoint x interrupt register (OTG_DOEPINTx)*

Table 586. Device-mode control and status registers (continued)

Acronym	Offset address	Register name
OTG_DOEPTSIZ0	0xB10	*Section 50.15.57: OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0)*
OTG_DOEPCTLx	0xB20 0xB40 ... 0xBA0	*Section 50.15.58: OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx)* *Section 50.15.59: OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx)*
OTG_DOEPTSIZx	0xB30 0xB50 ... 0xBB0	*Section 50.15.60: OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx)*

Data FIFO (DFIFO) access register map

These registers, available in both host and device modes, are used to read or write the FIFO space for a specific endpoint or a channel, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel.

Table 587. Data FIFO (DFIFO) access register map

FIFO access register section	Offset address	Access
Device IN endpoint 0/Host OUT Channel 0: DFIFO write access Device OUT endpoint 0/Host IN Channel 0: DFIFO read access	0x1000–0x1FFC	w r
Device IN endpoint 1/Host OUT Channel 1: DFIFO write access Device OUT endpoint 1/Host IN Channel 1: DFIFO read access	0x2000–0x2FFC	w r
...	...	...
Device IN endpoint x(1)/Host OUT Channel x(1): DFIFO write access Device OUT endpoint x(1)/Host IN Channel x(1): DFIFO read access	0xX000–0xXFFC	w r

1. Where x is 5in device mode and 11 in host mode.

Power and clock gating CSR map

There is a single register for power and clock gating. It is available in both host and device modes.

Table 588. Power and clock gating control and status registers

Acronym	Offset address	Register name
OTG_PCGCCTL	0xE00–0xE04	*Section 50.15.61: OTG power and clock gating control register (OTG_PCGCCTL)*

50.15 OTG_FS registers

These registers are available in both host and device modes, and do not need to be reprogrammed when switching between these modes.

Bit values in the register descriptions are expressed in binary unless otherwise specified.

50.15.1 OTG control and status register (OTG_GOTGCTL)

The OTG_GOTGCTL register controls the behavior and reflects the status of the OTG function of the core.

Address offset: 0x000

Reset value: 0b0000 0XXX XX00 X001 0000 0000 XX00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CUR MOD	OTG VER	BSVLD	ASVLD	DBCT	CID STS
										r	rw	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EHEN	Res.	Res.	Res.	Res.	BVALO VAL	BVALO EN	AVALO VAL	AVALO EN	VBVAL OVAL	VBVAL OEN	Res.	Res.
			rw					rw	rw	rw	rw	rw	rw		

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

Bit 21 **CURMOD:** Current mode of operation
Indicates the current mode (host or device).
0: Device mode
1: Host mode

Bit 20 **OTGVER:** OTG version
Selects the OTG revision.
0:OTG Version 1.3. OTG1.3 is obsolete for new product development.
1:OTG Version 2.0. In this version the core supports only data line pulsing for SRP.

Bit 19 **BSVLD:** B-session valid
Indicates the device mode transceiver status.
0: B-session is not valid.
1: B-session is valid.
In OTG mode, the user can use this bit to determine if the device is connected or disconnected.
Note: Only accessible in device mode.

Bit 18 **ASVLD:** A-session valid
Indicates the host mode transceiver status.
0: A-session is not valid
1: A-session is valid
Note: Only accessible in host mode.

Bit 17 **DBCT:** Long/short debounce time
Indicates the debounce time of a detected connection.
0: Long debounce time, used for physical connections (100 ms + 2.5 µs)
1: Short debounce time, used for soft connections (2.5 µs)
Note: Only accessible in host mode.

Bit 16 **CIDSTS:** Connector ID status
Indicates the connector ID status on a connect event.
0: The OTG_FS controller is in A-device mode
1: The OTG_FS controller is in B-device mode
Note: Accessible in both device and host modes.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **EHEN:** Embedded host enable
It is used to select between OTG A device state machine and embedded host state machine.
0: OTG A device state machine is selected
1: Embedded host state machine is selected

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **BVALOVAL:** B-peripheral session valid override value.
This bit is used to set override value for Bvalid signal when BVALOEN bit is set.
0: Bvalid value is '0' when BVALOEN = 1
1: Bvalid value is '1' when BVALOEN = 1
Note: Only accessible in device mode.

Bit 6 **BVALOEN:** B-peripheral session valid override enable.
This bit is used to enable/disable the software to override the Bvalid signal using the BVALOVAL bit.
0: Override is disabled and Bvalid signal from the respective PHY selected is used internally by the core
1: Internally Bvalid received from the PHY is overridden with BVALOVAL bit value
Note: Only accessible in device mode.

Bit 5 **AVALOVAL:** A-peripheral session valid override value.
This bit is used to set override value for Avalid signal when AVALOEN bit is set.
0: Avalid value is '0' when AVALOEN = 1
1: Avalid value is '1' when AVALOEN = 1
Note: Only accessible in host mode.

Bit 4 **AVALOEN:** A-peripheral session valid override enable.
This bit is used to enable/disable the software to override the Avalid signal using the AVALOVAL bit.
0: Override is disabled and Avalid signal from the respective PHY selected is used internally by the core
1: Internally Avalid received from the PHY is overridden with AVALOVAL bit value
Note: Only accessible in host mode.

Bit 3 **VBVALOVAL:** V_{BUS} valid override value.

This bit is used to set override value for vbusvalid signal when VBVALOEN bit is set.

0: vbusvalid value is '0' when VBVALOEN = 1

1: vbusvalid value is '1' when VBVALOEN = 1

Note: Only accessible in host mode.

Bit 2 **VBVALOEN:** V_{BUS} valid override enable.

This bit is used to enable/disable the software to override the vbusvalid signal using the VBVALOVAL bit.

0: Override is disabled and vbusvalid signal from the respective PHY selected is used internally by the core

1: Internally vbusvalid received from the PHY is overridden with VBVALOVAL bit value

Note: Only accessible in host mode.

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

50.15.2 OTG interrupt register (OTG_GOTGINT)

The application reads this register whenever there is an OTG interrupt and clears the bits in this register to clear the OTG interrupt.

Address offset: 0x004

Reset value: 0b0000 0000 000X 0000 0000 0000 0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADTO CHG	Res.	Res.
													rc_w1		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEDET	Res.	Res.
													rc_w1		

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **ADTOCHG:** A-device timeout change

The core sets this bit to indicate that the A-device has timed out while waiting for the B-device to connect.

Note: Accessible in both device and host modes.

Bits 17:3 Reserved, must be kept at reset value.

Bit 2 **SEDET:** Session end detected

The core sets this bit to indicate that the level of the voltage on V_{BUS} is no longer valid for a B-Peripheral session when V_{BUS} < 0.8 V.

Note: Accessible in both device and host modes.

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

50.15.3 OTG AHB configuration register (OTG_GAHBCFG)

This register can be used to configure the core after power-on or a change in mode. This register mainly contains AHB system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.

Address offset: 0x008

Reset value: 0x0000 0000

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PTXFELVL:** Periodic Tx FIFO empty level

Indicates when the periodic Tx FIFO empty interrupt bit in the OTG_GINTSTS register (PTXFE bit in OTG_GINTSTS) is triggered.

0: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is half empty

1: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is completely empty

Note: Only accessible in host mode.

Bit 7 **TXFELVL:** Tx FIFO empty level

Condition: device mode. This bit indicates when IN endpoint Transmit FIFO empty interrupt (TXFE in OTG_DIEPINTx) is triggered:

0: The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is half empty

1: The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is completely empty

Condition: host mode. This bit indicates when the nonperiodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) is triggered:

0: The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is half empty

1: The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is completely empty

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **GINTMSK:** Global interrupt mask

The application uses this bit to mask or unmask the interrupt line assertion to itself. Irrespective of this bit's setting, the interrupt status registers are updated by the core.

0: Mask the interrupt assertion to the application.

1: Unmask the interrupt assertion to the application.

Note: Accessible in both device and host modes.

50.15.4 OTG USB configuration register (OTG_GUSBCFG)

This register can be used to configure the core after power-on or a changing to host mode or device mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the AHB or the USB. Do not make changes to this register after the initial programming.

Address offset: 0x00C

Reset value: 0b0X00 0000 0000 0000 0001 0100 0100 0000

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **FDMOD:** Force device mode

Writing a 1 to this bit, forces the core to device mode irrespective of the OTG_ID input pin.

0: Normal mode

1: Force device mode

After setting the force bit, the application must wait at least 25 ms before the change takes effect.

Note: Accessible in both device and host modes.

Bit 29 **FHMOD:** Force host mode

Writing a 1 to this bit, forces the core to host mode irrespective of the OTG_ID input pin.

0: Normal mode

1: Force host mode

After setting the force bit, the application must wait at least 25 ms before the change takes effect.

Note: Accessible in both device and host modes.

Bits 28:14 Reserved, must be kept at reset value.

Bits 13:10 **TRDT[3:0]:** USB turnaround time

These bits allows to set the turnaround time in PHY clocks. They must be configured according to *Table 589: TRDT values*, depending on the application AHB frequency. Higher TRDT values allow stretching the USB response time to IN tokens in order to compensate for longer AHB read access latency to the data FIFO.

Note: Only accessible in device mode.

Bits 9:7 Reserved, must be kept at reset value.

Bit 6 **PHYSEL:** Full Speed serial transceiver mode select
This bit is always 1 with read-only access.

Bits 5:3 Reserved, must be kept at reset value.

Bits 2:0 **TOCAL[2:0]:** FS timeout calibration
The number of PHY clocks that the application programs in this field is added to the full-speed interpacket timeout duration in the core to account for any additional delays introduced by the PHY. This can be required, because the delay introduced by the PHY in generating the line state condition can vary from one PHY to another.
The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock is 0.25 bit times.

Table 589. TRDT values

AHB frequency range (MHz)		TRDT minimum value
Min	Max	
14.2	15	0xF
15	16	0xE
16	17.2	0xD
17.2	18.5	0xC
18.5	20	0xB
20	21.8	0xA
21.8	24	0x9
24	27.5	0x8
27.5	32	0x7
32	-	0x6

50.15.5 OTG reset register (OTG_GRSTCTL)

The application uses this register to reset various hardware features inside the core.

Address offset: 0x010

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHB IDL	Res.	CSRST DONE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r		rc_w1													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TXFNUM[4:0]					TXF FLSH	RXF FLSH	Res.	FCRST	PSRST	CSRST
					rw	rw	rw	rw	rw	rs	rs		rs	rs	rw

Bit 31 **AHBIDL:** AHB master idle

Indicates that the AHB master state machine is in the Idle condition.

Note: Accessible in both device and host modes.

Bit 30 Reserved, must be kept at reset value.

Bit 29 **CSRSTDONE**: Core soft reset done

The core sets this bit when all the necessary logic is reset in the core. This bit is cleared by the application along with CSRST.

0: Core reset not done

1: Core reset done

Bits 28:11 Reserved, must be kept at reset value.

Bits 10:6 **TXFNUM[4:0]:** Tx FIFO number

This is the FIFO number that must be flushed using the Tx FIFO Flush bit. This field must not be changed until the core clears the Tx FIFO Flush bit.

Condition: host mode

00000: Non-periodic Tx FIFO flush

00001: Periodic Tx FIFO flush

10000: Flush all the transmit FIFOs

Condition: device mode

00000: Tx FIFO 0 flush

00001: Tx FIFO 1 flush

00010: Tx FIFO 2 flush

...

01111: Tx FIFO 15 flush

10000: Flush all the transmit FIFOs

Note: Accessible in both device and host modes.

Bit 5 **TXFFLSH:** Tx FIFO flush

This bit selectively flushes a single or all transmit FIFOs, but cannot do so if the core is in the midst of a transaction.

The application must write this bit only after checking that the core is neither writing to the Tx FIFO nor reading from the Tx FIFO. Verify using these registers:

Read—NAK Effective interrupt ensures the core is not reading from the FIFO

Write—AHBIDL bit in OTG_GRSTCTL ensures the core is not writing anything to the FIFO.

Flushing is normally recommended when FIFOs are reconfigured. FIFO flushing is also recommended during device endpoint disable. The application must wait until the core clears this bit before performing any operations. This bit takes eight clocks to clear, using the slower clock of phy_clk or hclk.

Note: Accessible in both device and host modes.

Bit 4 **RXFFLSH:** Rx FIFO flush

The application can flush the entire Rx FIFO using this bit, but must first ensure that the core is not in the middle of a transaction.

The application must only write to this bit after checking that the core is neither reading from the Rx FIFO nor writing to the Rx FIFO.

The application must wait until the bit is cleared before performing any other operations. This bit requires 8 clocks (slowest of PHY or AHB clock) to clear.

Note: Accessible in both device and host modes.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FCRST:** Host frame counter reset

The application writes this bit to reset the frame number counter inside the core. When the frame counter is reset, the subsequent SOF sent out by the core has a frame number of 0.

When application writes '1' to the bit, it might not be able to read back the value as it gets cleared by the core in a few clock cycles.

Note: Only accessible in host mode.

Bit 1 **PSRST:** Partial soft reset

Resets the internal state machines but keeps the enumeration info. Could be used to recover some specific PHY errors.

Note: Accessible in both device and host modes.

Bit 0 **CSRST:** Core soft reset

Resets the HCLK and PHY clock domains as follows:

Clears the interrupts and all the CSR register bits except for the following bits:

– GATEHCLK bit in OTG_PCGCCTL
– STPPCLK bit in OTG_PCGCCTL
– FSLSPCS bits in OTG_HCFG
– DSPD bit in OTG_DCFG
– SDIS bit in OTG_DCTL
– OTG_GCCFG register

All module state machines (except for the AHB slave unit) are reset to the Idle state, and all the transmit FIFOs and the receive FIFO are flushed.

Any transactions on the AHB Master are terminated as soon as possible, after completing the last data phase of an AHB transfer. Any transactions on the USB are terminated immediately.

The application can write to this bit any time it wants to reset the core. The application must clear this bit after checking CSRSTDONE (core soft reset done). The application must also check that AHBIDL = 1 (AHB master is IDLE) before starting any operation.

Typically, the software reset is used during software development and also when the PHY selection bits are dynamically changed in the USB configuration registers listed above. When the PHY is changed, the corresponding clock for the PHY is selected and used in the PHY domain. Once a new clock is selected, the PHY domain has to be reset for proper operation.

Note: Accessible in both device and host modes.

50.15.6 OTG core interrupt register [alternate] (OTG_GINTSTS)

Valid for Host mode, see next section for Device mode.

This register also indicates the current mode. To clear the interrupt status bits of the rc_w1 type, the application must write 1 into the bit.

This register interrupts the application for system-level events in the current mode (device mode or host mode).

The FIFO status interrupts are read-only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the OTG_GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

Address offset: 0x014

Reset value: 0b0X0X 0100 0000 0000 0000 0000 0010 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WKUP INT	SRQ INT	DISC INT	CIDS CHG	LPM INT	PTXFE	HCINT	HPRT INT	RST DET	Res.	IPXFR	IISOI XFR	OEP INT	IEPINT	Res.	Res.
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	r	r	r	rc_w1		rc_w1	rc_w1	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPF	ISOO DRP	ENUM DNE	USB RST	USB SUSP	ESUSP	Res.	Res.	GO NAK EFF	GI NAK EFF	NPTXF E	RXF LVL	SOF	OTG INT	MMIS	CMOD
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1			r	r	r	r	rc_w1	r	rc_w1	r

Bit 31 **WKUPINT:** Resume/remote wakeup detected interrupt

Wakeup interrupt during suspend(L2) or LPM(L1) state.

– During suspend(L2):

In device mode, this interrupt is asserted when a resume is detected on the USB. In host mode, this interrupt is asserted when a remote wakeup is detected on the USB.

– During LPM(L1):

This interrupt is asserted for either host initiated resume or device initiated remote wakeup on USB.

Note: Accessible in both device and host modes.

Bit 30 **SRQINT:** Session request/new session detected interrupt

In host mode, this interrupt is asserted when a session request is detected from the device. In device mode, this interrupt is asserted when V_{BUS} is in the valid range for a B-peripheral device. Accessible in both device and host modes.

Bit 29 **DISCINT:** Disconnect detected interrupt

Asserted when a device disconnect is detected.

Note: Only accessible in host mode.

Bit 28 **CIDSCHG:** Connector ID status change

The core sets this bit when there is a change in connector ID status.

Note: Accessible in both device and host modes.

Bit 27 **LPMINT:** LPM interrupt

In device mode, this interrupt is asserted when the device receives an LPM transaction and responds with a non-ERRORed response.

In host mode, this interrupt is asserted when the device responds to an LPM transaction with a non-ERRORed response or when the host core has completed LPM transactions for the programmed number of times (RETRYCNT bit in OTG_GLPMCFG).

This field is valid only if the LPMEN bit in OTG_GLPMCFG is set to 1.

Bit 26 **PTXFE:** Periodic Tx FIFO empty

Asserted when the periodic transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the periodic request queue. The half or completely empty status is determined by the periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (PTXFELVL bit in OTG_GAHBCFG).

Note: Only accessible in host mode.

Bit 25 **HCINT:** Host channels interrupt

The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in host mode). The application must read the OTG_HAINT register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding OTG_HCINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the OTG_HCINTx register to clear this bit.

Note: Only accessible in host mode.

Bit 24 **HPRTINT:** Host port interrupt

The core sets this bit to indicate a change in port status of one of the OTG_FS controller ports in host mode. The application must read the OTG_HPRT register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_HPRT register to clear this bit.

Note: Only accessible in host mode.

Bit 23 **RSTDET:** Reset detected interrupt

In device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in suspend.

Note: Only accessible in device mode.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **IPXFR:** Incomplete periodic transfer

In host mode, the core sets this interrupt bit when there are incomplete periodic transactions still pending, which are scheduled for the current frame.

Bit 20 **IISOIXFR:** Incomplete isochronous IN transfer

The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Note: Only accessible in device mode.

Bit 19 **OEPINT:** OUT endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding OTG_DOEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DOEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bit 18 **IEPINT:** IN endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding OTG_DIEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DIEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPF:** End of periodic frame interrupt

Indicates that the period specified in the periodic frame interval field of the OTG_DCFG register (PFIVL bit in OTG_DCFG) has been reached in the current frame.

Note: Only accessible in device mode.

Bit 14 **ISOODRP:** Isochronous OUT packet dropped interrupt

The core sets this bit when it fails to write an isochronous OUT packet into the Rx FIFO because the Rx FIFO does not have enough space to accommodate a maximum size packet for the isochronous OUT endpoint.

Note: Only accessible in device mode.

Bit 13 **ENUMDNE:** Enumeration done

The core sets this bit to indicate that speed enumeration is complete. The application must read the OTG_DSTS register to obtain the enumerated speed.

Note: Only accessible in device mode.

Bit 12 **USBRST:** USB reset

The core sets this bit to indicate that a reset is detected on the USB.

Note: Only accessible in device mode.

Bit 11 **USBSUSP:** USB suspend

The core sets this bit to indicate that a suspend was detected on the USB. The core enters the suspended state when there is no activity on the data lines for an extended period of time.

Note: Only accessible in device mode.

Bit 10 **ESUSP:** Early suspend

The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.

Note: Only accessible in device mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFF:** Global OUT NAK effective

Indicates that the Set global OUT NAK bit in the OTG_DCTL register (SGONAK bit in OTG_DCTL), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear global OUT NAK bit in the OTG_DCTL register (CGONAK bit in OTG_DCTL).

Note: Only accessible in device mode.

Bit 6 **GINAKEFF:** Global IN non-periodic NAK effective

Indicates that the Set global non-periodic IN NAK bit in the OTG_DCTL register (SGINAK bit in OTG_DCTL), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear global non-periodic IN NAK bit in the OTG_DCTL register (CGINAK bit in OTG_DCTL).

This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.

Note: Only accessible in device mode.

Bit 5 **NPTXFE:** Non-periodic Tx FIFO empty

This interrupt is asserted when the non-periodic Tx FIFO is either half or completely empty, and there is space for at least one entry to be written to the non-periodic transmit request queue. The half or completely empty status is determined by the non-periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

Note: Accessible in host mode only.

Bit 4 **RXFLVL:** Rx FIFO non-empty

Indicates that there is at least one packet pending to be read from the Rx FIFO.

Note: Accessible in both host and device modes.

Bit 3 **SOF:** Start of frame

In host mode, the core sets this bit to indicate that an SOF (FS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt.

In device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the OTG_DSTS register to get the current frame number. This interrupt is seen only when the core is operating in FS.

Note: This register may return '1' if read immediately after power on reset. If the register bit reads '1' immediately after power on reset it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.

Note: Accessible in both host and device modes.

Bit 2 **OTGINT:** OTG interrupt

The core sets this bit to indicate an OTG protocol event. The application must read the OTG interrupt status (OTG_GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_GOTGINT register to clear this bit.

Note: Accessible in both host and device modes.

Bit 1 **MMIS:** Mode mismatch interrupt

The core sets this bit when the application is trying to access:

– A host mode register, when the core is operating in device mode

– A device mode register, when the core is operating in host mode

The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.

Note: Accessible in both host and device modes.

Bit 0 **CMOD:** Current mode of operation

Indicates the current mode.

0: Device mode

1: Host mode

Note: Accessible in both host and device modes.

50.15.7 OTG core interrupt register [alternate] (OTG_GINTSTS)

Valid for Device mode, see previous section for Host mode.

This register also indicates the current mode. To clear the interrupt status bits of the rc_w1 type, the application must write 1 into the bit.

This register interrupts the application for system-level events in the current mode (device mode or host mode).

The FIFO status interrupts are read-only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the OTG_GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

Address offset: 0x014

Reset value: 0x0400 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WKUP INT	SRQ INT	DISC INT	CIDS CHG	LPM INT	PTXFE	HCINT	HPRT INT	RST DET	Res.	IN COMP ISO OUT	IISOI XFR	OEP INT	IEPINT	Res.	Res.
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	r	r	r	rc_w1		rc_w1	rc_w1	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPF	ISOO DRP	ENUM DNE	USB RST	USB SUSP	ESUSP	Res.	Res.	GO NAK EFF	GI NAK EFF	NPTXF E	RXF LVL	SOF	OTG INT	MMIS	CMOD
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1			r	r	r	r	rc_w1	r	rc_w1	r

Bit 31 **WKUPINT:** Resume/remote wakeup detected interrupt

Wakeup interrupt during suspend(L2) or LPM(L1) state.

– During suspend(L2):

In device mode, this interrupt is asserted when a resume is detected on the USB. In host mode, this interrupt is asserted when a remote wakeup is detected on the USB.

– During LPM(L1):

This interrupt is asserted for either host initiated resume or device initiated remote wakeup on USB.

Note: Accessible in both device and host modes.

Bit 30 **SRQINT:** Session request/new session detected interrupt

In host mode, this interrupt is asserted when a session request is detected from the device. In device mode, this interrupt is asserted when V_{BUS} is in the valid range for a B-peripheral device. Accessible in both device and host modes.

Bit 29 **DISCINT:** Disconnect detected interrupt

Asserted when a device disconnect is detected.

Note: Only accessible in host mode.

Bit 28 **CIDSCHG:** Connector ID status change

The core sets this bit when there is a change in connector ID status.

Note: Accessible in both device and host modes.

Bit 27 **LPMINT:** LPM interrupt

In device mode, this interrupt is asserted when the device receives an LPM transaction and responds with a non-ERRORed response.

In host mode, this interrupt is asserted when the device responds to an LPM transaction with a non-ERRORed response or when the host core has completed LPM transactions for the programmed number of times (RETRYCNT bit in OTG_GLPMCFG).

This field is valid only if the LPMEN bit in OTG_GLPMCFG is set to 1.

Bit 26 **PTXFE:** Periodic Tx FIFO empty

Asserted when the periodic transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the periodic request queue. The half or completely empty status is determined by the periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (PTXFELVL bit in OTG_GAHBCFG).

Note: Only accessible in host mode.

Bit 25 **HCINT:** Host channels interrupt

The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in host mode). The application must read the OTG_HAINT register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding OTG_HCINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the OTG_HCINTx register to clear this bit.

Note: Only accessible in host mode.

Bit 24 **HPRTINT:** Host port interrupt

The core sets this bit to indicate a change in port status of one of the OTG_FS controller ports in host mode. The application must read the OTG_HPRT register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_HPRT register to clear this bit.

Note: Only accessible in host mode.

Bit 23 **RSTDET:** Reset detected interrupt

In device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in suspend.

Note: Only accessible in device mode.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **INCOMPISOOUT:** Incomplete isochronous OUT transfer

In device mode, the core sets this interrupt to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Bit 20 **IISOIXFR:** Incomplete isochronous IN transfer

The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Note: Only accessible in device mode.

Bit 19 **OEPINT:** OUT endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding OTG_DOEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DOEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bit 18 **IEPINT:** IN endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding OTG_DIEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DIEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPF:** End of periodic frame interrupt

Indicates that the period specified in the periodic frame interval field of the OTG_DCFG register (PFIVL bit in OTG_DCFG) has been reached in the current frame.

Note: Only accessible in device mode.

Bit 14 **ISOODRP:** Isochronous OUT packet dropped interrupt

The core sets this bit when it fails to write an isochronous OUT packet into the Rx FIFO because the Rx FIFO does not have enough space to accommodate a maximum size packet for the isochronous OUT endpoint.

Note: Only accessible in device mode.

Bit 13 **ENUMDNE:** Enumeration done

The core sets this bit to indicate that speed enumeration is complete. The application must read the OTG_DSTS register to obtain the enumerated speed.

Note: Only accessible in device mode.

Bit 12 **USBRST:** USB reset

The core sets this bit to indicate that a reset is detected on the USB.

Note: Only accessible in device mode.

Bit 11 **USBSUSP:** USB suspend

The core sets this bit to indicate that a suspend was detected on the USB. The core enters the suspended state when there is no activity on the data lines for an extended period of time.

Note: Only accessible in device mode.

Bit 10 **ESUSP:** Early suspend

The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.

Note: Only accessible in device mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFF:** Global OUT NAK effective

Indicates that the Set global OUT NAK bit in the OTG_DCTL register (SGONAK bit in OTG_DCTL), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear global OUT NAK bit in the OTG_DCTL register (CGONAK bit in OTG_DCTL).

Note: Only accessible in device mode.

Bit 6 **GINAKEFF:** Global IN non-periodic NAK effective

Indicates that the Set global non-periodic IN NAK bit in the OTG_DCTL register (SGINAK bit in OTG_DCTL), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear global non-periodic IN NAK bit in the OTG_DCTL register (CGINAK bit in OTG_DCTL).

This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.

Note: Only accessible in device mode.

Bit 5 **NPTXFE:** Non-periodic Tx FIFO empty

This interrupt is asserted when the non-periodic Tx FIFO is either half or completely empty, and there is space for at least one entry to be written to the non-periodic transmit request queue. The half or completely empty status is determined by the non-periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

Note: Accessible in host mode only.

Bit 4 **RXFLVL:** Rx FIFO non-empty

Indicates that there is at least one packet pending to be read from the Rx FIFO.

Note: Accessible in both host and device modes.

Bit 3 **SOF:** Start of frame

In host mode, the core sets this bit to indicate that an SOF (FS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt.

In device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the OTG_DSTS register to get the current frame number. This interrupt is seen only when the core is operating in FS.

Note: This register may return '1' if read immediately after power on reset. If the register bit reads '1' immediately after power on reset it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.

Note: Accessible in both host and device modes.

Bit 2 **OTGINT:** OTG interrupt

The core sets this bit to indicate an OTG protocol event. The application must read the OTG interrupt status (OTG_GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_GOTGINT register to clear this bit.

Note: Accessible in both host and device modes.

Bit 1 **MMIS:** Mode mismatch interrupt

The core sets this bit when the application is trying to access:

– A host mode register, when the core is operating in device mode

– A device mode register, when the core is operating in host mode

The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.

Note: Accessible in both host and device modes.

Bit 0 **CMOD:** Current mode of operation

Indicates the current mode.

0: Device mode

1: Host mode

Note: Accessible in both host and device modes.

50.15.8 OTG interrupt mask register [alternate] (OTG_GINTMSK)

Valid for Host mode, see next section for Device mode.

This register works with the core interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the core interrupt (OTG_GINTSTS) register bit corresponding to that interrupt is still set.

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUIM	SRQIM	DISCINT	CIDSCHGM	LPMINTM	PTXFEM	HCIM	PRTIM	Res.	Res.	IPXFRM	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw			rc_w1					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Res.
										rw	rw	rw	rw	rw	

Bit 31 **WUIM:** Resume/remote wakeup detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 30 **SRQIM:** Session request/new session detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 29 **DISCINT:** Disconnect detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 28 **CIDSCHGM:** Connector ID status change mask
0: Masked interrupt
1: Unmasked interrupt

Bit 27 **LPMINTM:** LPM interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 26 **PTXFEM:** Periodic Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 25 **HCIM:** Host channels interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 24 **PRTIM:** Host port interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **IPXFRM:** Incomplete periodic transfer mask
0: Masked interrupt
1: Unmasked interrupt

Bits 20:6 Reserved, must be kept at reset value.

Bit 5 **NPTXFEM:** Non-periodic Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **RXFLVLM:** Receive FIFO non-empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **SOFM:** Start of frame mask
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **OTGINT:** OTG interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **MMISM:** Mode mismatch interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 Reserved, must be kept at reset value.

50.15.9 OTG interrupt mask register [alternate] (OTG_GINTMSK)

Valid for Device mode, see previous section for Host mode.

This register works with the core interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the core interrupt (OTG_GINTSTS) register bit corresponding to that interrupt is still set.

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUIM	SRQIM	Res.	CIDSCHGM	LPMINTM	Res.	Res.	Res.	RSTDETM	Res.	IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	Res.	Res.
rw	rw		rw	rw				rw		rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Res.	Res.	GONAKEFFM	GINAKEFFM	Res.	RXFLVLM	SOFM	OTGINT	MMISM	Res.
rw	rw	rw	rw	rw	rw			rw	rw		rw	rw	rw	rw	

Bit 31 **WUIM:** Resume/remote wakeup detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and device modes.

Bit 30 **SRQIM:** Session request/new session detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and device modes.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **CIDSCHGM:** Connector ID status change mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and device modes.

Bit 27 **LPMINTM:** LPM interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and device modes.

Bits 26:24 Reserved, must be kept at reset value.

Bit 23 **RSTDETM:** Reset detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 22 Reserved, must be kept at reset value.

Bit 21 **IISOOXFRM:** Incomplete isochronous OUT transfer mask
0: Masked interrupt
1: Unmasked interrupt

Bit 20 **IISOIXFRM:** Incomplete isochronous IN transfer mask
0: Masked interrupt
1: Unmasked interrupt

Bit 19 **OEPINT:** OUT endpoints interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 18 **IEPINT:** IN endpoints interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPFM:** End of periodic frame interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 14 **ISOODRPM:** Isochronous OUT packet dropped interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 13 **ENUMDNEM:** Enumeration done mask
0: Masked interrupt
1: Unmasked interrupt

Bit 12 **USBRST:** USB reset mask
0: Masked interrupt
1: Unmasked interrupt

Bit 11 **USBSUSPM:** USB suspend mask
0: Masked interrupt
1: Unmasked interrupt

Bit 10 **ESUSPM:** Early suspend mask
0: Masked interrupt
1: Unmasked interrupt

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFFM:** Global OUT NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 6 **GINAKEFFM:** Global non-periodic IN NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 5 Reserved, must be kept at reset value.

Bit 4 **RXFLVLM:** Receive FIFO non-empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **SOFM:** Start of frame mask
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **OTGINT:** OTG interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **MMISM:** Mode mismatch interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 Reserved, must be kept at reset value.

50.15.10 OTG receive status debug read [alternate] (OTG_GRXSTSR)

Valid for Host mode, see next section for Device mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS[3:0]				DPID
											r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID	BCNT[10:0]											CHNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **PKTSTS[3:0]:** Packet status

Indicates the status of the received packet
0010: IN data packet received
0011: IN transfer completed (triggers an interrupt)
0101: Data toggle error (triggers an interrupt)
0111: Channel halted (triggers an interrupt)
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID

Indicates the data PID of the received packet
00: DATA0
10: DATA1

Bits 14:4 **BCNT[10:0]:** Byte count

Indicates the byte count of the received IN data packet.

Bits 3:0 **CHNUM[3:0]:** Channel number

Indicates the channel number to which the current received packet belongs.

50.15.11 OTG receive status debug read register [alternate] (OTG_GRXSTSR)

Valid for Device mode, see previous section for Host mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM[3:0]				PKTSTS[3:0]				DPID[1]
				r			r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID[0]	BCNT[10:0]											EPNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **STSPHST:** Status phase start

Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern.

Bits 26:25 Reserved, must be kept at reset value.

Bits 24:21 **FRMNUM[3:0]:** Frame number

This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Bits 20:17 **PKTSTS[3:0]:** Packet status
Indicates the status of the received packet
0001: Global OUT NAK (triggers an interrupt)
0010: OUT data packet received
0011: OUT transfer completed (triggers an interrupt)
0100: SETUP transaction completed (triggers an interrupt)
0110: SETUP data packet received
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID
Indicates the data PID of the received OUT data packet
00: DATA0
10: DATA1

Bits 14:4 **BCNT[10:0]:** Byte count
Indicates the byte count of the received data packet.

Bits 3:0 **EPNUM[3:0]:** Endpoint number
Indicates the endpoint number to which the current received packet belongs.

50.15.12 OTG status read and pop registers [alternate] (OTG_GRXSTSP)

Valid for Host mode, see next section for Device mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS[3:0]				DPID
											r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID	BCNT[10:0]											CHNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **PKTSTS[3:0]:** Packet status

Indicates the status of the received packet
0010: IN data packet received
0011: IN transfer completed (triggers an interrupt)
0101: Data toggle error (triggers an interrupt)
0111: Channel halted (triggers an interrupt)
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID

Indicates the data PID of the received packet
00: DATA0
10: DATA1

Bits 14:4 **BCNT[10:0]:** Byte count

Indicates the byte count of the received IN data packet.

Bits 3:0 **CHNUM[3:0]:** Channel number

Indicates the channel number to which the current received packet belongs.

50.15.13 OTG status read and pop registers [alternate] (OTG_GRXSTSP)

Valid for Device mode, see previous section for Host mode.

This description is for register OTG_GRXSTSP in Device mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

Address offset: 0x020

Reset value: 0x0000 0000

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

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **STSPHST:** Status phase start

Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern.

Bits 26:25 Reserved, must be kept at reset value.

Bits 24:21 **FRMNUM[3:0]:** Frame number
This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Bits 20:17 **PKTSTS[3:0]:** Packet status
Indicates the status of the received packet
0001: Global OUT NAK (triggers an interrupt)
0010: OUT data packet received
0011: OUT transfer completed (triggers an interrupt)
0100: SETUP transaction completed (triggers an interrupt)
0110: SETUP data packet received
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID
Indicates the data PID of the received OUT data packet
00: DATA0
10: DATA1

Bits 14:4 **BCNT[10:0]:** Byte count
Indicates the byte count of the received data packet.

Bits 3:0 **EPNUM[3:0]:** Endpoint number
Indicates the endpoint number to which the current received packet belongs.

50.15.14 OTG receive FIFO size register (OTG_GRXFSIZ)

The application can program the RAM size that must be allocated to the Rx FIFO.

Address offset: 0x024

Reset value: 0x0000 0200

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RXFD[15:0]:** Rx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

50.15.15 OTG host non-periodic transmit FIFO size register [alternate] (OTG_HNPTXFSIZ)

Valid for Host mode, see next section for Device mode.

Address offset: 0x028

Reset value: 0x0200 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPTXFD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPTXFSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **NPTXFD[15:0]:** Non-periodic Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **NPTXFSA[15:0]:** Non-periodic transmit RAM start address
This field configures the memory start address for non-periodic transmit FIFO RAM.

50.15.16 Endpoint 0 Transmit FIFO size [alternate] (OTG_DIEPTXF0)

Valid for Device mode, see previous section for Host mode.

Address offset: 0x028

Reset value: 0x0200 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TX0FD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX0FSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **TX0FD[15:0]:** Endpoint 0 Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **TX0FSA[15:0]:** Endpoint 0 transmit RAM start address
This field configures the memory start address for the endpoint 0 transmit FIFO RAM.

50.15.17 OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)

Host mode only, this register is not valid in Device mode.

This read-only register contains the free space information for the non-periodic Tx FIFO and the non-periodic transmit request queue.

Address offset: 0x02C

Reset value: 0x0008 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	NPTXQTOP[6:0]							NPTQXSAV[7:0]							
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPTXFSAV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **NPTXQTOP[6:0]:** Top of the non-periodic transmit request queue
Entry in the non-periodic Tx request queue that is currently being processed by the MAC.
Bits 30:27: Channel/endpoint number
Bits 26:25:
XXXX00X: IN/OUT token
XXXX01X: Zero-length transmit packet (device IN/host OUT)
XXXX11X: Channel halt command
Bit 24: Terminate (last entry for selected channel/endpoint)

Bits 23:16 **NPTQXSAV[7:0]:** Non-periodic transmit request queue space available
Indicates the amount of free space available in the non-periodic transmit request queue. This queue holds both IN and OUT requests.
0: Non-periodic transmit request queue is full
1: 1 location available
2: locations available
n: n locations available (0 ≤ n ≤ 8)
Others: Reserved

Bits 15:0 **NPTXFSAV[15:0]:** Non-periodic Tx FIFO space available
Indicates the amount of free space available in the non-periodic Tx FIFO.
Values are in terms of 32-bit words.
0: Non-periodic Tx FIFO is full
1: 1 word available
2: 2 words available
n: n words available (where 0 ≤ n ≤ 512)
Others: Reserved

50.15.18 OTG general core configuration register (OTG_GCCFG)

Address offset: 0x038

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDPULL UPEN	Res.	Res.	FORCE HOST PD	Res.	Res.	Res.	VBDEN	SDEN	PDEN	Res.	BCDEN	PWR DWN
			rw			rw				rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SESS VLD	PS2 DET	SDET	PDET	Res.
											r	r	r	r	

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDPULLUPEN:** ID pin pull-up enable
0: Disable ID pin pull-up
1: Enable ID pin pull-up

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **FORCEHOSTPD:** Force host mode pull-downs
If the ID pin functions are enabled, the host mode pull-downs on DP and DM activate automatically. However, whenever that is not the case, yet host mode is required, this bit must be used to force the pull-downs active.
0: Do not force host mode pull-downs
1: Force host mode pull-downs

Bit 24 Reserved, must be kept at reset value.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **VBDEN:** USB V_{BUS} detection enable
Enables V_{BUS} sensing comparators to detect V_{BUS} valid levels on the V_{BUS} PAD for USB host and device operation. If HNP and/or SRP support is enabled, V_{BUS} comparators are automatically enabled independently of VBDEN value.
0 = V_{BUS} detection disabled
1 = V_{BUS} detection enabled

Bit 20 **SDEN:** Secondary detection (SD) mode enable
This bit is set by the software to put the BCD into SD mode. Only one detection mode (PD, SD or OFF) must be selected to work correctly

Bit 19 **PDEN:** Primary detection (PD) mode enable
This bit is set by the software to put the BCD into PD mode. Only one detection mode (PD, SD or OFF) must be selected to work correctly.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **BCDEN:** Battery charging detector (BCD) enable
This bit is set by the software to enable the BCD support within the USB device. When enabled, the USB PHY is fully controlled by BCD and cannot be used for normal communication. Once the BCD discovery is finished, the BCD must be placed in OFF mode by clearing this bit to ‘0’ in order to allow the normal USB operation.

Bit 16 **PWRDWN:** Power down control of FS PHY

Used to activate the FS PHY in transmission/reception. When reset, the PHY is kept in power-down. When set, the BCD function must be off (BCDEN=0).

0 = USB FS PHY disabled

1 = USB FS PHY enabled

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **SESSVLD:** VBUS session indicator

Indicates if VBUS is above VBUS session threshold.

0: VBUS is below VBUS session threshold

1: VBUS is above VBUS session threshold

Bit 3 **PS2DET:** DM pull-up detection status

This bit is active only during PD and gives the result of comparison between DM voltage level and VLGC threshold. In normal situation, the DM level must be below this threshold. If it is above, it means that the DM is externally pulled high. This can be caused by connection to a PS2 port (which pulls-up both DP and DM lines) or to some proprietary charger not following the BCD specification.

0: Normal port detected (connected to SDP, CDP or DCP)

1: PS2 port or proprietary charger detected

Bit 2 **SDET:** Secondary detection (SD) status

This bit gives the result of SD.

0: CDP detected

1: DCP detected

Bit 1 **PDET:** Primary detection (PD) status

This bit gives the result of PD.

0: no BCD support detected (connected to SDP or proprietary device).

1: BCD support detected (connected to CDP or DCP).

Bit 0 Reserved, must be kept at reset value.

50.15.19 OTG core ID register (OTG_CID)

This is a register containing the Product ID as reset value.

Address offset: 0x03C

Reset value: 0x0000 5200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRODUCT_ID[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRODUCT_ID[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRODUCT_ID[31:0]:** Product ID field

Application-programmable ID field.

50.15.20 OTG core LPM configuration register (OTG_GLPMCFG)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EN BESL	LPMRCNTSTS[2:0]			SND LPM	LPMRCNT[2:0]			LPMCHIDX[3:0]				L1RSM OK
			rw	r	r	r	rs	rw	rw	rw	rw	rw	rw	rw	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLP STS	LPMRSP[1:0]		L1DS EN	BESLTHRS[3:0]				L1SS EN	REM WAKE	BESL[3:0]				LPM ACK	LPM EN
r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ENBESL:** Enable best effort service latency

This bit enables the BESL feature as defined in the LPM errata:

0: The core works as described in the following document:
USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007

1: The core works as described in the LPM Errata:
Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007

Note: Only the updated behavior (described in LPM Errata) is considered in this document and so the ENBESL bit must be set to '1' by application SW.

Bits 27:25 **LPMRCNTSTS[2:0]:** LPM retry count status

Number of LPM host retries still remaining to be transmitted for the current LPM sequence.

Note: Accessible only in host mode.

Bit 24 **SNDLPM:** Send LPM transaction

When the application software sets this bit, an LPM transaction containing two tokens, EXT and LPM is sent. The hardware clears this bit once a valid response (STALL, NYET, or ACK) is received from the device or the core has finished transmitting the programmed number of LPM retries.

Note: This bit must be set only when the host is connected to a local port.

Note: Accessible only in host mode.

Bits 23:21 **LPMRCNT[2:0]:** LPM retry count

When the device gives an ERROR response, this is the number of additional LPM retries that the host performs until a valid device response (STALL, NYET, or ACK) is received.

Note: Accessible only in host mode.

Bits 20:17 **LPMCHIDX[3:0]:** LPM Channel Index

The channel number on which the LPM transaction has to be applied while sending an LPM transaction to the local device. Based on the LPM channel index, the core automatically inserts the device address and endpoint number programmed in the corresponding channel into the LPM transaction.

Note: Accessible only in host mode.

Bit 16 **L1RSMOK:** Sleep state resume OK

Indicates that the device or host can start resume from Sleep state. This bit is valid in LPM sleep (L1) state. It is set in sleep mode after a delay of 50 µs ($T_{L1Residency}$).
This bit is reset when SLPSTS is 0.
1: The application or host can start resume from Sleep state
0: The application or host cannot start resume from Sleep state

Bit 15 **SLPSTS:** Port sleep status

Device mode:
This bit is set as long as a Sleep condition is present on the USB bus. The core enters the Sleep state when an ACK response is sent to an LPM transaction and the $T_{L1TokenRetry}$ timer has expired. To stop the PHY clock, the application must set the STPPCLK bit in OTG_PCGCCTL, which asserts the PHY suspend input signal.
The application must rely on SLPSTS and not ACK in LPMRSP to confirm transition into sleep.
The core comes out of sleep:
– When there is any activity on the USB linestate
– When the application writes to the RWUSIG bit in OTG_DCTL or when the application resets or soft-disconnects the device.

Host mode:
The host transitions to Sleep (L1) state as a side-effect of a successful LPM transaction by the core to the local port with ACK response from the device. The read value of this bit reflects the current Sleep status of the port.
The core clears this bit after:
– The core detects a remote L1 wakeup signal,
– The application sets the PRST bit or the PRES bit in the OTG_HPRT register, or
– The application sets the L1Resume/ remote wakeup detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT bit in OTG_GINTSTS, respectively).
0: Core not in L1
1: Core in L1

Bits 14:13 **LPMRSP[1:0]:** LPM response

Device mode:
The response of the core to LPM transaction received is reflected in these two bits.
Host mode:
Handshake response received from local device for LPM transaction
11: ACK
10: NYET
01: STALL
00: ERROR (No handshake response)

Bit 12 **L1DSEN:** L1 deep sleep enable

Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit must be set to '1' by application SW in all the cases.

Bits 11:8 **BESLTHRS[3:0]:** BESL threshold

Device mode:

The core puts the PHY into deep low power mode in L1 when BESL value is greater than or equal to the value defined in this field BESL_Thres[3:0].

Host mode:

The core puts the PHY into deep low power mode in L1. BESLTHRS[3:0] specifies the time for which resume signaling is to be reflected by host ($T_{L1HubDrvResume2}$) on the USB bus when it detects device initiated resume.

BESLTHRS must not be programmed with a value greater than 1100b in host mode, because this exceeds maximum $T_{L1HubDrvResume2}$.

Thres[3:0] host mode resume signaling time (µs):

0000: 75

0001: 100

0010: 150

0011: 250

0100: 350

0101: 450

0110: 950

All other values: reserved

Bit 7 **L1SSEN:** L1 Shallow Sleep enable

Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit must be set to '1' by application SW in all the cases.

Bit 6 **REMWAKE:** bRemoteWake value

Host mode:

The value of remote wake up to be sent in the wIndex field of LPM transaction.

Device mode (read-only):

This field is updated with the received LPM token bRemoteWake bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.

Bits 5:2 **BESL[3:0]:** Best effort service latency

Host mode:

The value of BESL to be sent in an LPM transaction. This value is also used to initiate resume for a duration $T_{L1HubDrvResume1}$ for host initiated resume.

Device mode (read-only):

This field is updated with the received LPM token BESL bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.

BESL[3:0]T_{BESL} (µs)

0000: 125
0001: 150
0010: 200
0011: 300
0100: 400
0101: 500
0110: 1000
0111: 2000
1000: 3000
1001: 4000
1010: 5000
1011: 6000
1100: 7000
1101: 8000
1110: 9000
1111: 10000

Bit 1 **LPMACK:** LPM token acknowledge enable

Handshake response to LPM token preprogrammed by device application software.

1: ACK

Even though ACK is preprogrammed, the core device responds with ACK only on successful LPM transaction. The LPM transaction is successful if:

– No PID/CRC5 errors in either EXT token or LPM token (else ERROR)
– Valid bLinkState = 0001B (L1) received in LPM transaction (else STALL)
– No data pending in transmit queue (else NYET).

0: NYET

The preprogrammed software bit is over-ridden for response to LPM token when:

– The received bLinkState is not L1 (STALL response), or
– An error is detected in either of the LPM token packets because of corruption (ERROR response).

Note: Accessible only in device mode.

Bit 0 **LPMEN:** LPM support enable

The application uses this bit to control the OTG_FS core LPM capabilities.

If the core operates as a non-LPM-capable host, it cannot request the connected device or hub to activate LPM mode.

If the core operates as a non-LPM-capable device, it cannot respond to any LPM transactions.

0: LPM capability is not enabled

1: LPM capability is enabled

50.15.21 OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)

Address offset: 0x100

Reset value: 0x0200 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTXFSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **PTXFSIZ[15:0]:** Host periodic Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16

Bits 15:0 **PTXSA[15:0]:** Host periodic Tx FIFO start address
This field configures the memory start address for periodic transmit FIFO RAM.

50.15.22 OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx)

Address offset: 0x104 + 0x04 * (x - 1), (x = 1 to 5)

Reset value: 0x0200 0400, 0x0200 0600, 0x0200 0800, 0x0200 0A00, 0x0200 0C00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INEPTXFD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTXSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **INEPTXFD[15:0]:** IN endpoint Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16

Bits 15:0 **INEPTXSA[15:0]:** IN endpoint FIFOx transmit RAM start address
This field contains the memory start address for IN endpoint transmit FIFOx. The address must be aligned with a 32-bit memory location.

50.15.23 Host-mode registers

Bit values in the register descriptions are expressed in binary unless otherwise specified.

Host-mode registers affect the operation of the core in the host mode. Host mode registers must not be accessed in device mode, as the results are undefined. Host mode registers can be categorized as follows:

50.15.24 OTG host configuration register (OTG_HCFG)

This register configures the core after power-on. Do not make changes to this register after initializing the host.

Address offset: 0x400

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSLSS	FSLSPCS[1:0]	
													r	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FSLSS:** FS- and LS-only support

The application uses this bit to control the core's enumeration speed. Using this bit, the application can make the core enumerate as an FS host, even if the connected device supports HS traffic. Do not make changes to this field after initial programming.
1: FS/LS-only, even if the connected device can support HS (read-only).

Bits 1:0 **FSLSPCS[1:0]:** FS/LS PHY clock select

Condition: FS Host mode
01: PHY clock is running at 48 MHz
Others: Reserved
Condition: LS Host mode
00: Reserved
01: Select 48 MHz PHY clock frequency
10: Select 6 MHz PHY clock frequency
11: Reserved

Note: The FSLSPCS must be set on a connection event according to the speed of the connected device (after changing this bit, a software reset must be performed).

50.15.25 OTG host frame interval register (OTG_HFIR)

This register stores the frame interval information for the current speed to which the OTG_FS controller has enumerated.

Address offset: 0x404

Reset value: 0x0000 EA60

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLD CTRL
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRIVL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RLDCTRL:** Reload control

This bit allows dynamic reloading of the HFIR register during run time.

0: The HFIR cannot be reloaded dynamically

1: The HFIR can be dynamically reloaded during run time.

This bit needs to be programmed during initial configuration and its value must not be changed during run time.

Caution: RLDCTRL = 0 is not recommended.

Bits 15:0 **FRIVL[15:0]:** Frame interval

The value that the application programs to this field, specifies the interval between two consecutive SOFs (FS) or Keep-Alive tokens (LS). This field contains the number of PHY clocks that constitute the required frame interval. The application can write a value to this register only after the port enable bit of the host port control and status register (PENA bit in OTG_HPRT) has been set. If no value is programmed, the core calculates the value based on the PHY clock specified in the FS/LS PHY clock select field of the host configuration register (FSLSPCS in OTG_HCFG). Do not change the value of this field after the initial configuration, unless the RLDCTRL bit is set. In such case, the FRIVL is reloaded with each SOF event.

– Frame interval = 1 ms × (FRIVL - 1)

50.15.26 OTG host frame number/frame time remaining register (OTG_HFNUM)

This register indicates the current frame number. It also indicates the time remaining (in terms of the number of PHY clocks) in the current frame.

Address offset: 0x408

Reset value: 0x0000 3FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FTREM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRNUM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **FTREM[15:0]:** Frame time remaining

Indicates the amount of time remaining in the current frame, in terms of PHY clocks. This field decrements on each PHY clock. When it reaches zero, this field is reloaded with the value in the Frame interval register and a new SOF is transmitted on the USB.

Bits 15:0 **FRNUM[15:0]:** Frame number

This field increments when a new SOF is transmitted on the USB, and is cleared to 0 when it reaches 0x3FFF.

50.15.27 OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)

This read-only register contains the free space information for the periodic Tx FIFO and the periodic transmit request queue.

Address offset: 0x410

Reset value: 0x0008 0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTXQTOP[7:0]								PTXQSAV[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXFSAVL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **PTXQTOP[7:0]:** Top of the periodic transmit request queue

This indicates the entry in the periodic Tx request queue that is currently being processed by the MAC.
This register is used for debugging.
Bit 31: Odd/Even frame
0XXXXXXX: send in even frame
1XXXXXXX: send in odd frame
Bits 30:27: Channel/endpoint number
Bits 26:25: Type
XXXXX00X: IN/OUT
XXXXX01X: Zero-length packet
XXXXX11X: Disable channel command
Bit 24: Terminate (last entry for the selected channel/endpoint)

Bits 23:16 **PTXQSAV[7:0]:** Periodic transmit request queue space available

Indicates the number of free locations available to be written in the periodic transmit request queue. This queue holds both IN and OUT requests.
00: Periodic transmit request queue is full
01: 1 location available
10: 2 locations available
n: n locations available (0 ≤ n ≤ 8)
Others: Reserved

Bits 15:0 **PTXFSAVL[15:0]:** Periodic transmit data FIFO space available

Indicates the number of free locations available to be written to in the periodic Tx FIFO.
Values are in terms of 32-bit words
0000: Periodic Tx FIFO is full
0001: 1 word available
0010: 2 words available
n: n words available (where 0 ≤ n ≤ PTXFD)
Others: Reserved

50.15.28 OTG host all channels interrupt register (OTG_HAINT)

When a significant event occurs on a channel, the host all channels interrupt register interrupts the application using the host channels interrupt bit of the core interrupt register (HCINT bit in OTG_GINTSTS). This is shown in *Figure 1018*. There is one interrupt bit per channel, up to a maximum of 16 bits. Bits in this register are set and cleared when the application sets and clears bits in the corresponding host channel-x interrupt register.

Address offset: 0x414

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HAINT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **HAINT[15:0]:** Channel interrupts
One bit per channel: Bit 0 for Channel 0, bit 15 for Channel 15

50.15.29 OTG host all channels interrupt mask register (OTG_HAINTMSK)

The host all channel interrupt mask register works with the host all channel interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per channel, up to a maximum of 16 bits.

Address offset: 0x418

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HAINTM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **HAINTM[15:0]:** Channel interrupt mask
0: Masked interrupt
1: Unmasked interrupt
One bit per channel: Bit 0 for channel 0, bit 15 for channel 15

50.15.30 OTG host port control and status register (OTG_HPRT)

This register is available only in host mode. Currently, the OTG host supports only one port.

A single register holds USB port-related information such as USB reset, enable, suspend, resume, connect status, and test mode for each port. It is shown in *Figure 1018*. The rc_w1 bits in this register can trigger an interrupt to the application through the host port interrupt bit of the core interrupt register (HPRTINT bit in OTG_GINTSTS). On a port interrupt, the application must read this register and clear the bit that caused the interrupt. For the rc_w1 bits, the application must write a 1 to the bit to clear the interrupt.

Address offset: 0x440

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSPD[1:0]		PTCTL[3]
													r	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTCTL[2:0]			PPWR	PLSTS[1:0]		Res.	PRST	PSUSP	PRES	POC CHNG	POCA	PEN CHNG	PENA	PCDET	PCSTS
rw	rw	rw	rw	r	r		rw	rs	rw	rc_w1	r	rc_w1	rc_w1	rc_w1	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 **PSPD[1:0]:** Port speed

Indicates the speed of the device attached to this port.
01: Full speed
10: Low speed
11: Reserved

Bits 16:13 **PTCTL[3:0]:** Port test control

The application writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is signaled on the port.
0000: Test mode disabled
0001: Test_J mode
0010: Test_K mode
0011: Test_SE0_NAK mode
0100: Test_Packet mode
0101: Test_Force_Enable
Others: Reserved

Bit 12 **PPWR:** Port power

The application uses this field to control power to this port, and the core clears this bit on an overcurrent condition.
0: Power off
1: Power on

Bits 11:10 **PLSTS[1:0]:** Port line status

Indicates the current logic level USB data lines
Bit 10: Logic level of OTG_DP
Bit 11: Logic level of OTG_DM

Bit 9 Reserved, must be kept at reset value.

Bit 8 **PRST:** Port reset

When the application sets this bit, a reset sequence is started on this port. The application must time the reset period and clear this bit after the reset sequence is complete.

0: Port not in reset

1: Port in reset

The application must leave this bit set for a minimum duration of at least 10 ms to start a reset on the port. The application can leave it set for another 10 ms in addition to the required minimum duration, before clearing the bit, even though there is no maximum limit set by the USB standard.

High speed: 50 ms

Full speed/Low speed: 10 ms

Bit 7 **PSUSP:** Port suspend

The application sets this bit to put this port in suspend mode. The core only stops sending SOFs when this is set. To stop the PHY clock, the application must set the port clock stop bit, which asserts the suspend input pin of the PHY.

The read value of this bit reflects the current suspend status of the port. This bit is cleared by the core after a remote wakeup signal is detected or the application sets the port reset bit or port resume bit in this register or the resume/remote wakeup detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT in OTG_GINTSTS, respectively).

0: Port not in suspend mode

1: Port in suspend mode

Bit 6 **PRES:** Port resume

The application sets this bit to drive resume signaling on the port. The core continues to drive the resume signal until the application clears this bit.

If the core detects a USB remote wakeup sequence, as indicated by the port resume/remote wakeup detected interrupt bit of the core interrupt register (WKUPINT bit in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit when it detects a disconnect condition. The read value of this bit indicates whether the core is currently driving resume signaling.

0: No resume driven

1: Resume driven

When LPM is enabled and the core is in L1 state, the behavior of this bit is as follow:

1. The application sets this bit to drive resume signaling on the port.

2. The core continues to drive the resume signal until a predetermined time specified in BESLTHRS[3:0] field of OTG_GLPMCFG register.

3. If the core detects a USB remote wakeup sequence, as indicated by the port L1Resume/Remote L1Wakeup detected interrupt bit of the core interrupt register (WKUPINT in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit at the end of resume.This bit can be set or cleared by both the core and the application. This bit is cleared by the core even if there is no device connected to the host.

Bit 5 **POCCHNG:** Port overcurrent change

The core sets this bit when the status of the port overcurrent active bit (bit 4) in this register changes.

Bit 4 **POCA:** Port overcurrent active

Indicates the overcurrent condition of the port.

0: No overcurrent condition

1: Overcurrent condition

Bit 3 **PENCHNG:** Port enable/disable change

The core sets this bit when the status of the port enable bit 2 in this register changes.

Bit 2 **PENA:** Port enable

A port is enabled only by the core after a reset sequence, and is disabled by an overcurrent condition, a disconnect condition, or by the application clearing this bit. The application cannot set this bit by a register write. It can only clear it to disable the port. This bit does not trigger any interrupt to the application.

0: Port disabled

1: Port enabled

Bit 1 **PCDET:** Port connect detected

The core sets this bit when a device connection is detected to trigger an interrupt to the application using the host port interrupt bit in the core interrupt register (HPRTINT bit in OTG_GINTSTS). The application must write a 1 to this bit to clear the interrupt.

Bit 0 **PCSTS:** Port connect status

0: No device is attached to the port

1: A device is attached to the port

50.15.31 OTG host channel x characteristics register (OTG_HCCHARx)

Address offset: 0x500 + 0x20 * x, (x = 0 to 11)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CHENA	CHDIS	ODD FRM	DAD[6:0]							MCNT[1:0]		EPTYP[1:0]		LSDEV	Res.
rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPDIR	EPNUM[3:0]				MPSIZ[10:0]										
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CHENA:** Channel enable

This field is set by the application and cleared by the OTG host.

0: Channel disabled

1: Channel enabled

Bit 30 **CHDIS:** Channel disable

The application sets this bit to stop transmitting/receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel disabled interrupt before treating the channel as disabled.

Bit 29 **ODDFRM:** Odd frame

This field is set (reset) by the application to indicate that the OTG host must perform a transfer in an odd frame. This field is applicable for only periodic (isochronous and interrupt) transactions.

0: Even frame

1: Odd frame

Bits 28:22 **DAD[6:0]:** Device address

This field selects the specific device serving as the data source or sink.

Bits 21:20 **MCNT[1:0]:** Multicount

This field indicates to the host the number of transactions that must be executed per frame for this periodic endpoint. For non-periodic transfers, this field is not used

00: Reserved. This field yields undefined results

01: 1 transaction

10: 2 transactions per frame to be issued for this endpoint

11: 3 transactions per frame to be issued for this endpoint

Note: This field must be set to at least 01.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

Indicates the transfer type selected.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

Bit 17 **LSDEV:** Low-speed device

This field is set by the application to indicate that this channel is communicating to a low-speed device.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **EPDIR:** Endpoint direction

Indicates whether the transaction is IN or OUT.

0: OUT

1: IN

Bits 14:11 **EPNUM[3:0]:** Endpoint number

Indicates the endpoint number on the device serving as the data source or sink.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

Indicates the maximum packet size of the associated endpoint.

50.15.32 OTG host channel x interrupt register (OTG_HCINTx)

This register indicates the status of a channel with respect to USB- and AHB-related events. It is shown in *Figure 1018*. The application must read this register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the host all channels interrupt (OTG_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers.

Address offset: 0x508 + 0x20 * x, (x = 0 to 11)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DTERR	FRM OR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
					rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DTERR:** Data toggle error.

Bit 9 **FRMOR:** Frame overrun.

Bit 8 **BBERR:** Babble error.

Bit 7 **TXERR:** Transaction error.

Indicates one of the following errors occurred on the USB.
CRC check failure
Timeout
Bit stuff error
False EOP

Bit 6 Reserved, must be kept at reset value.

Bit 5 **ACK:** ACK response received/transmitted interrupt.

Bit 4 **NAK:** NAK response received interrupt.

Bit 3 **STALL:** STALL response received interrupt.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CHH:** Channel halted.

Indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application.

Bit 0 **XFRC:** Transfer completed.

Transfer completed normally without any errors.

50.15.33 OTG host channel x interrupt mask register (OTG_HCINTMSKx)

This register reflects the mask for each channel status described in the previous section.

Address offset: 0x50C + 0x20 * x, (x = 0 to 11)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	Res.	ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
					rw	rw	rw	rw		rw	rw	rw		rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DTERRM:** Data toggle error mask.

0: Masked interrupt
1: Unmasked interrupt

Bit 9 **FRMORM:** Frame overrun mask.

0: Masked interrupt
1: Unmasked interrupt

Bit 8 **BBERRM:** Babble error mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 7 **TXERRM:** Transaction error mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 6 Reserved, must be kept at reset value.

Bit 5 **ACKM:** ACK response received/transmitted interrupt mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **NAKM:** NAK response received interrupt mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STALLM:** STALL response received interrupt mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CHHM:** Channel halted mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed mask
0: Masked interrupt
1: Unmasked interrupt

50.15.34 OTG host channel x transfer size register (OTG_HCTSIZx)

Address offset: 0x510 + 0x20 * x, (x = 0 to 11)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DO PNG	DPID[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XFRSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DOPNG:** Do Ping

This bit is used only for OUT transfers. Setting this field to 1 directs the host to do PING protocol.

Note: Do not set this bit for IN transfers. If this bit is set for IN transfers, it disables the channel.

Bits 30:29 **DPID[1:0]:** Data PID

The application programs this field with the type of PID to use for the initial transaction. The host maintains this field for the rest of the transfer.

00: DATA0

10: DATA1

11: SETUP (control) / reserved (non-control)

Bits 28:19 **PKTCNT[9:0]:** Packet count

This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN).

The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size

For an OUT, this field is the number of data bytes the host sends during the transfer.

For an IN, this field is the buffer size that the application has reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).

50.15.35 Device-mode registers

These registers must be programmed every time the core changes to device mode

50.15.36 OTG device configuration register (OTG_DCFG)

This register configures the core in device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

Address offset: 0x800

Reset value: 0x0220 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRATIM	Res.	Res.	PFIVL[1:0]		DAD[6:0]							Res.	NZLSOHSK	DSPD[1:0]	
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **ERRATIM:** Erratic error interrupt mask

1: Mask early suspend interrupt on erratic error

0: Early suspend interrupt is generated on erratic error

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bits 12:11 **PFIVL[1:0]:** Periodic frame interval

Indicates the time within a frame at which the application must be notified using the end of periodic frame interrupt. This can be used to determine if all the isochronous traffic for that frame is complete.

00: 80% of the frame interval

01: 85% of the frame interval

10: 90% of the frame interval

11: 95% of the frame interval

Bits 10:4 **DAD[6:0]:** Device address

The application must program this field after every SetAddress control command.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **NZLSOHSK:** Non-zero-length status OUT handshake

The application can use this field to select the handshake the core sends on receiving a nonzero-length data packet during the OUT transaction of a control transfer's status stage.

1: Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application.

0: Send the received OUT packet to the application (zero-length or nonzero-length) and send a handshake based on the NAK and STALL bits for the endpoint in the device endpoint control register.

Bits 1:0 **DSPD[1:0]:** Device speed

Indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected.

00: Reserved

01: Reserved

10: Reserved

11: Full speed (USB 1.1 transceiver clock is 48 MHz)

50.15.37 OTG device control register (OTG_DCTL)

Address offset: 0x804

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DS BESL RJCT	Res.	Res.
													rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PO PRG DNE	CGO NAK	SGO NAK	CGI NAK	SGI NAK	TCTL[2:0]			GON STS	GIN STS	SDIS	RWU SIG
				rw	w	w	w	w	rw	rw	rw	r	r	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **DSBESLRJCT:** Deep sleep BESL reject

Core rejects LPM request with BESL value greater than BESL threshold programmed. NYET response is sent for LPM tokens with BESL value greater than BESL threshold. By default, the deep sleep BESL reject feature is disabled.

Bits 17:12 Reserved, must be kept at reset value.

Bit 11 **POPRGDNE:** Power-on programming done

The application uses this bit to indicate that register programming is completed after a wakeup from power down mode.

Bit 10 **CGONAK:** Clear global OUT NAK

Writing 1 to this field clears the Global OUT NAK.

Bit 9 **SGONAK:** Set global OUT NAK

Writing 1 to this field sets the Global OUT NAK.
The application uses this bit to send a NAK handshake on all OUT endpoints.
The application must set the this bit only after making sure that the Global OUT NAK effective bit in the core interrupt register (GONAKEFF bit in OTG_GINTSTS) is cleared.

Bit 8 **CGINAK:** Clear global IN NAK

Writing 1 to this field clears the Global IN NAK.

Bit 7 **SGINAK:** Set global IN NAK

Writing 1 to this field sets the Global non-periodic IN NAK.The application uses this bit to send a NAK handshake on all non-periodic IN endpoints.
The application must set this bit only after making sure that the Global IN NAK effective bit in the core interrupt register (GINAKEFF bit in OTG_GINTSTS) is cleared.

Bits 6:4 **TCTL[2:0]:** Test control

000: Test mode disabled
001: Test_J mode
010: Test_K mode
011: Test_SE0_NAK mode
100: Test_Packet mode
101: Test_Force_Enable
Others: Reserved

Bit 3 **GONSTS:** Global OUT NAK status

0: A handshake is sent based on the FIFO status and the NAK and STALL bit settings.
1: No data is written to the Rx FIFO, irrespective of space availability. Sends a NAK handshake on all packets, except on SETUP transactions. All isochronous OUT packets are dropped.

Bit 2 **GINSTS:** Global IN NAK status

0: A handshake is sent out based on the data availability in the transmit FIFO.

1: A NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO.

Bit 1 **SDIS:** Soft disconnect

The application uses this bit to signal the USB OTG core to perform a soft disconnect. As long as this bit is set, the host does not see that the device is connected, and the device does not receive signals on the USB. The core stays in the disconnected state until the application clears this bit.

0: Normal operation. When this bit is cleared after a soft disconnect, the core generates a device connect event to the USB host. When the device is reconnected, the USB host restarts device enumeration.

1: The core generates a device disconnect event to the USB host.

Bit 0 **RWUSIG:** Remote wakeup signaling

When the application sets this bit, the core initiates remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the suspend state. As specified in the USB 2.0 specification, the application must clear this bit 1 ms to 15 ms after setting it.

If LPM is enabled and the core is in the L1 (sleep) state, when the application sets this bit, the core initiates L1 remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the sleep state. As specified in the LPM specification, the hardware automatically clears this bit 50 µs ($T_{L1DevDrvResume}$) after being set by the application. The application must not set this bit when bRemoteWake from the previous LPM transaction is zero (refer to REMWAKE bit in GLPMCFG register).

Table 590 contains the minimum duration (according to device state) for which the Soft disconnect (SDIS) bit must be set for the USB host to detect a device disconnect. To accommodate clock jitter, it is recommended that the application add some extra delay to the specified minimum duration.

Table 590. Minimum duration for soft disconnect

Operating speed	Device state	Minimum duration
Full speed	Suspended	1 ms + 2.5 µs
Full speed	Idle	2.5 µs
Full speed	Not Idle or suspended (Performing transactions)	2.5 µs

50.15.38 OTG device status register (OTG_DSTS)

This register indicates the status of the core with respect to USB-related events. It must be read on interrupts from the device all interrupts (OTG_DAINT) register.

Address offset: 0x808

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVLNSTS[1:0]		FNSOF[13:8]					
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FNSOF[7:0]								Res.	Res.	Res.	Res.	EERR	ENUMSPD[1:0]		SUSP STS
r	r	r	r	r	r	r	r					r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **DEVLNSTS[1:0]:** Device line status

Indicates the current logic level USB data lines.
Bit [23]: Logic level of D+
Bit [22]: Logic level of D-

Bits 21:8 **FNSOF[13:0]:** Frame number of the received SOF

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **EERR:** Erratic error

The core sets this bit to report any erratic errors.
Due to erratic errors, the OTG_FS controller goes into suspended state and an interrupt is generated to the application with Early suspend bit of the OTG_GINTSTS register (ESUSP bit in OTG_GINTSTS). If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover.

Bits 2:1 **ENUMSPD[1:0]:** Enumerated speed

Indicates the speed at which the OTG_FS controller has come up after speed detection through a chirp sequence.
11: Full speed using embedded FS PHY
Others: reserved

Bit 0 **SUSPSTS:** Suspend status

In device mode, this bit is set as long as a suspend condition is detected on the USB. The core enters the suspended state when there is no activity on the USB data lines for a period of 3 ms. The core comes out of the suspend:

– When there is an activity on the USB data lines
– When the application writes to the remote wakeup signaling bit in the OTG_DCTL register (RWUSIG bit in OTG_DCTL).

50.15.39 OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)

This register works with each of the OTG_DIEPINTx registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the OTG_DIEPINTx register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.

Address offset: 0x810

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAKM	Res.	Res.	Res.	Res.	Res.	Res.	INEPNEM	INEPNMM	ITTXFEMSK	TOM	Res.	EPDM	XFRCM
		rw							rw	rw	rw	rw		rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAKM:** NAK interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 12:7 Reserved, must be kept at reset value.

Bit 6 **INEPNEM:** IN endpoint NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 5 **INEPNMM:** IN token received with EP mismatch mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **ITTXFEMSK:** IN token received when Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **TOM:** Timeout condition mask (Non-isochronous endpoints)
0: Masked interrupt
1: Unmasked interrupt

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

50.15.40 OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK)

This register works with each of the OTG_DOEPINTx registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the OTG_DOEPINTx register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

Address offset: 0x814

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAK MSK	BERR M	Res.	Res.	Res.	OUT PKT ERRM	Res.	Res.	STS PHSR XM	OTEPD M	STUPM	Res.	EPDM	XFRC M
		rw	rw				rw			rw	rw	rw		rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAKMSK:** NAK interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 12 **BERRM**: Babble error interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **OUTPKTERRM:** Out packet error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **STSPHSRXM**: Status phase received for control write mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **OTEPDM:** OUT token received when endpoint disabled mask. Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STUPM:** SETUP phase done mask. Applies to control endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

50.15.41 OTG device all endpoints interrupt register (OTG_DAINT)

When a significant event occurs on an endpoint, a OTG_DAINT register interrupts the application using the device OUT endpoints interrupt bit or device IN endpoints interrupt bit of the OTG_GINTSTS register (OEPINT or IEPINT in OTG_GINTSTS, respectively). There is one interrupt bit per endpoint, up to a maximum of 16 bits for OUT endpoints and 16 bits for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used. Bits in this register are set and cleared when the application sets and clears bits in the corresponding device endpoint-x interrupt register (OTG_DIEPINTx/OTG_DOEPINTx).

Address offset: 0x818

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEPINT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IEPINT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **OEPINT[15:0]:** OUT endpoint interrupt bits
One bit per OUT endpoint:
Bit 16 for OUT endpoint 0, bit 19 for OUT endpoint 3.

Bits 15:0 **IEPINT[15:0]:** IN endpoint interrupt bits
One bit per IN endpoint:
Bit 0 for IN endpoint 0, bit 3 for endpoint 3.

50.15.42 OTG all endpoints interrupt mask register (OTG_DAINTMSK)

The OTG_DAINTMSK register works with the device endpoint interrupt register to interrupt the application when an event occurs on a device endpoint. However, the OTG_DAINT register bit corresponding to that interrupt is still set.

Address offset: 0x81C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEPM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IEPM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **OEPM[15:0]:** OUT EP interrupt mask bits
One per OUT endpoint:
Bit 16 for OUT EP 0, bit 19 for OUT EP 3
0: Masked interrupt
1: Unmasked interrupt

Bits 15:0 **IEPM[15:0]:** IN EP interrupt mask bits
One bit per IN endpoint:
Bit 0 for IN EP 0, bit 3 for IN EP 3
0: Masked interrupt
1: Unmasked interrupt

50.15.43 OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK)

This register is used to control the IN endpoint FIFO empty interrupt generation (TXFE_OTG_DIEPINTx).

Address offset: 0x834

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTXFEM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INEPTXFEM[15:0]:** IN EP Tx FIFO empty interrupt mask bits
These bits act as mask bits for OTG_DIEPINTx.
TXFE interrupt one bit per IN endpoint:
Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3
0: Masked interrupt
1: Unmasked interrupt

50.15.44 OTG device each endpoint interrupt register (OTG_DEACHINT)

Address offset: 0x0838

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OEP1INT	Res.
														r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IEP1INT	Res.
														r	

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **OEP1INT:** OUT endpoint 1 interrupt bit

Bits 16:2 Reserved, must be kept at reset value.

Bit 1 **IEP1INT:** IN endpoint 1interrupt bit

Bit 0 Reserved, must be kept at reset value.

50.15.45 OTG device each endpoint interrupt mask register (OTG_DEACHINTMSK)

There is one interrupt bit for endpoint 1 IN and one interrupt bit for endpoint 1 OUT.

Address offset: 0x083C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OEP1INTM	Res.
														rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IEP1INTM	Res.
														rw	

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **OEP1INTM:** OUT endpoint 1 interrupt mask bit

Bits 16:2 Reserved, must be kept at reset value.

Bit 1 **IEP1INTM:** IN endpoint 1 interrupt mask bit

Bit 0 Reserved, must be kept at reset value.

50.15.46 OTG device each IN endpoint-1 interrupt mask register (OTG_DIEPEACHM)

This register works with the OTG_DIEPINT1 register to generate a dedicated interrupt OTG_EP1_IN for endpoint #1. The IN endpoint interrupt for a specific status in the OTG_DOEPINT1 register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

Address offset: 0x844

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAKM	Res.	Res.	Res.	Res.	TXFURM	Res.	INEPNEM	Res.	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM
		rw					rw		rw		rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAKM:** NAK interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **TXFURM:** FIFO underrun mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **INEPNEM:** IN endpoint NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 5 Reserved, must be kept at reset value.

Bit 4 **ITTXFEMSK:** IN token received when Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **TOM:** Timeout condition mask (Non-isochronous endpoints)
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM:** AHB error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

50.15.47 OTG device each OUT endpoint-1 interrupt mask register (OTG_DOEPEACHMSK1)

This register works with the OTG_DOEPINT1 register to generate a dedicated interrupt OTG_EP1_OUT for endpoint #1. The OUT endpoint interrupt for a specific status in the OTG_DOEPINT1 register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

Address offset: 0x884

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NYET MSK	NAK MSK	BERR M	Res.	Res.	Res.	OUT PKT ERRM	Res.	Res.	Res.	OTEPD M	STUPM	AHB ERRM	EPDM	XFRC M
	rw	rw	rw				rw				rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **NYETMSK:** NYET interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 13 **NAKMSK:** NAK interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 12 **BERRM**: Babble error interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **OUTPKTERRM:** Out packet error mask
0: Masked interrupt
1: Unmasked interrupt

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **OTEPDM:** OUT token received when endpoint disabled mask
Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STUPM:** SETUP phase done mask
Applies to control endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM:** AHB error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

50.15.48 OTG device control IN endpoint 0 control register (OTG_DIEPCTL0)

This section describes the OTG_DIEPCTL0 register for USB_OTG FS. Nonzero control endpoints use registers for endpoints 1–3.

Address offset: 0x900

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	Res.	Res.	SNAK	CNAK	TXFNUM[3:0]				STALL	Res.	EPTYP[1:0]		NAK STS	Res.
rs	rs			w	w	rw	rw	rw	rw	rs		r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ[1:0]	
r														rw	rw

Bit 31 **EPENA:** Endpoint enable
The application sets this bit to start transmitting data on the endpoint 0.
The core clears this bit before setting any of the following interrupts on this endpoint:
– Endpoint disabled
– Transfer completed

Bit 30 **EPDIS:** Endpoint disable
The application sets this bit to stop transmitting data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.

Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for an endpoint after a SETUP packet is received on that endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 **TXFNUM[3:0]:** Tx FIFO number

This value is set to the FIFO number that is assigned to IN endpoint 0.

Bit 21 **STALL:** STALL handshake

The application can only set this bit, and the core clears it when a SETUP token is received for this endpoint. If a NAK bit, a Global IN NAK or Global OUT NAK is set along with this bit, the STALL bit takes priority.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

Hardcoded to ‘00’ for control.

Bit 17 **NAKSTS:** NAK status

Indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status

1: The core is transmitting NAK handshakes on this endpoint.

When this bit is set, either by the application or core, the core stops transmitting data, even if there are data available in the Tx FIFO. Irrespective of this bit’s setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **USBAEP:** USB active endpoint

This bit is always set to 1, indicating that control endpoint 0 is always active in all configurations and interfaces.

Bits 14:2 Reserved, must be kept at reset value.

Bits 1:0 **MPSIZ[1:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint.

00: 64 bytes

01: 32 bytes

10: 16 bytes

11: 8 bytes

50.15.49 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx)

Valid for INT/BULK endpoints, see next section for ISO endpoints.

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0x900 + 0x20 * x, (x = 1 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SD1 PID	SD0 PID	SNAK	CNAK	TXFNUM[3:0]				STALL	Res.	EPTYP[1:0]		NAK STS	DPID
rs	rs	w	w	w	w	rw	rw	rw	rw	rw		rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable

The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
– SETUP phase done
– Endpoint disabled
– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SD1PID:** Set DATA1 PID

Writing to this field sets the endpoint data PID (DPID) field in this register to DATA1.

Bit 28 **SD0PID:** Set DATA0 PID

Applies to interrupt/bulk IN endpoints only.
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 **TXFNUM[3:0]:** Tx FIFO number

These bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.
This field is valid only for IN endpoints.

Bit 21 **STALL:** STALL handshake

Applies to non-control, non-isochronous IN endpoints only (access type is rw).

The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

This is the transfer type supported by this logical endpoint.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

Bit 17 **NAKSTS:** NAK status

It indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there are data available in the Tx FIFO.

For isochronous IN endpoints: The core sends out a zero-length data packet, even if there are data available in the Tx FIFO.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **DPID:** Endpoint data PID

Applies to interrupt/bulk IN endpoints only.

Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.

0: DATA0

1: DATA1

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

50.15.50 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx)

Valid for ISO endpoints, see previous section for INT/BULK endpoints.

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0x900 + 0x20 * x, (x = 1 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SODD FRM	SEVN FRM	SNAK	CNAK	TXFNUM[3:0]				STALL	Res.	EPTYP[1:0]		NAK STS	EO NUM
rs	rs	w	w	w	w	rw	rw	rw	rw	rw		rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable

The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
– SETUP phase done
– Endpoint disabled
– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SODDFRM:** Set odd frame

Applies to isochronous IN and OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 **SEVNFRM:** Set even frame

Applies to isochronous IN endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 **TXFNUM[3:0]:** Tx FIFO number

These bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.
This field is valid only for IN endpoints.

Bit 21 **STALL:** STALL handshake

Applies to non-control, non-isochronous IN endpoints only (access type is rw).

The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

This is the transfer type supported by this logical endpoint.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

Bit 17 **NAKSTS:** NAK status

It indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there are data available in the Tx FIFO.

For isochronous IN endpoints: The core sends out a zero-length data packet, even if there are data available in the Tx FIFO.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM:** Even/odd frame

Applies to isochronous IN endpoints only.

Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.

0: Even frame

1: Odd frame

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

50.15.51 OTG device IN endpoint x interrupt register (OTG_DIEPINTx)

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 1018*. The application must read this register when the IN endpoints interrupt bit of the core interrupt register (IEPINT in OTG_GINTSTS) is set. Before the application can read this register, it must first read the device all endpoints interrupt (OTG_DAINT) register to get the exact endpoint number for the device endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

Address offset: 0x908 + 0x20 * x, (x = 0 to 5)

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAK	Res.	PKTD RPSTS	Res.	Res.	Res.	TXFE	IN EPNE	IN EPNM	ITTXFE	TOC	Res.	EP DISD	XFRC
		rc_w1		rc_w1				r	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAK:** NAK input

The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **PKTDRPSTS:** Packet dropped status

This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **TXFE:** Transmit FIFO empty

This interrupt is asserted when the Tx FIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the Tx FIFO Empty Level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

Bit 6 **INEPNE:** IN endpoint NAK effective

This bit can be cleared when the application clears the IN endpoint NAK by writing to the CNAK bit in OTG_DIEPCTLx.

This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core.

This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.

Bit 5 **INEPNM**: IN token received with EP mismatch

Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 4 **ITTXFE:** IN token received when Tx FIFO is empty
Indicates that an IN token was received when the associated Tx FIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 3 **TOC:** Timeout condition
Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EPDISD:** Endpoint disabled interrupt
This bit indicates that the endpoint is disabled per the application's request.

Bit 0 **XFRC:** Transfer completed interrupt
This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

50.15.52 OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0)

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the device control endpoint 0 control registers (EPENA in OTG_DIEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–3.

Address offset: 0x910

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT[1:0]		Res.	Res.	Res.
											rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:19 **PKTCNT[1:0]:** Packet count
Indicates the total number of USB packets that constitute the transfer size amount of data for endpoint 0.
This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ[6:0]:** Transfer size
Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

50.15.53 OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx)

This read-only register contains the free space information for the device IN endpoint Tx FIFO.

Address offset: 0x918 + 0x20 * x, (x = 0 to 5)

Reset value: 0x0000 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTFSAV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INEPTFSAV[15:0]:** IN endpoint Tx FIFO space available
Indicates the amount of free space available in the endpoint Tx FIFO.
Values are in terms of 32-bit words:
0: Endpoint Tx FIFO is full
1: 1 word available
2: 2 words available
n: n words available
Others: Reserved

50.15.54 OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using the endpoint enable bit in the OTG_DIEPCTLx registers (EPENA bit in OTG_DIEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Address offset: 0x910 + 0x20 * x, (x = 1 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCNT[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XFRSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **MCNT[1:0]:** Multi count

For periodic IN endpoints, this field indicates the number of packets that must be transmitted per frame on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints.

01: 1 packet

10: 2 packets

11: 3 packets

Bits 28:19 **PKTCNT[9:0]:** Packet count

Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.

This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size

This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

50.15.55 OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)

This section describes the OTG_DOEPCTL0 register. Nonzero control endpoints use registers for endpoints 1–3.

Address offset: 0xB00

Reset value: 0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	Res.	Res.	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP[1:0]		NAK STS	Res.
w	r			w	w					rs	rw	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ[1:0]	
r														r	r

Bit 31 **EPENA:** Endpoint enable

The application sets this bit to start transmitting data on endpoint 0.

The core clears this bit before setting any of the following interrupts on this endpoint:

– SETUP phase done

– Endpoint disabled

– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application cannot disable control OUT endpoint 0.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.

Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake

The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

Hardcoded to 2'b00 for control.

Bit 17 **NAKSTS:** NAK status

Indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit, the core stops receiving data, even if there is space in the Rx FIFO to accommodate the incoming packet. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **USBAEP:** USB active endpoint

This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.

Bits 14:2 Reserved, must be kept at reset value.

Bits 1:0 **MPSIZ[1:0]:** Maximum packet size

The maximum packet size for control OUT endpoint 0 is the same as what is programmed in control IN endpoint 0.

00: 64 bytes

01: 32 bytes

10: 16 bytes

11: 8 bytes

50.15.56 OTG device OUT endpoint x interrupt register (OTG_DOEPINTx)

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 1018*. The application must read this register when the OUT endpoints interrupt bit of the OTG_GINTSTS register (OEPINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the OTG_DAINT register to get the exact endpoint number for the OTG_DOEPINTx register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

Address offset: 0xB08 + 0x20 * x, (x = 0 to 5)

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAK	BERR	Res.	Res.	Res.	Res.	Res.	Res.	STSPH SRX	OTEP DIS	STUP	Res.	EP DISD	XFRC
		rc_w1	rc_w1							rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAK:** NAK input

The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.

Bit 12 **BERR:** Babble error interrupt

The core generates this interrupt when babble is received for the endpoint.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **STSPHSRX**: Status phase received for control write

This interrupt is valid only for control OUT endpoints. This interrupt is generated only after OTG_FS has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a control write transfer. The application can use this interrupt to ACK or STALL the status phase, after it has decoded the data phase.

Bit 4 **OTEPDIS:** OUT token received when endpoint disabled

Applies only to control OUT endpoints.

Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.

Bit 3 **STUP:** SETUP phase done

Applies to control OUT endpoint only.Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EPDISD:** Endpoint disabled interrupt

This bit indicates that the endpoint is disabled per the application's request.

Bit 0 **XFRC:** Transfer completed interrupt

This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

50.15.57 OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0)

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the OTG_DOEPCTL0 registers (EPENA bit in OTG_DOEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–5.

Address offset: 0xB10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	STUPCNT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT	Res.	Res.	Res.
	rw	rw										rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **STUPCNT[1:0]:** SETUP packet count

This field specifies the number of back-to-back SETUP data packets the endpoint can receive.

01: 1 packet

10: 2 packets

11: 3 packets

Bits 28:20 Reserved, must be kept at reset value.

Bit 19 **PKTCNT:** Packet count

This field is decremented to zero after a packet is written into the Rx FIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ[6:0]:** Transfer size

Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.

50.15.58 OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx)

Valid for INT/BULK endpoints, see next section for ISO endpoints.

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0xB00 + 0x20 * x, (x = 1 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SD1 PID	SD0 PID	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP[1:0]		NAK STS	DPID
rs	rs	w	w	w	w					rw	rw	rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable

Applies to IN and OUT endpoints.

The application sets this bit to start transmitting data on an endpoint.

The core clears this bit before setting any of the following interrupts on this endpoint:

– SETUP phase done

– Endpoint disabled

– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SD1PID:** Set DATA1 PID

Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the endpoint data PID (DPID) field in this register to DATA1.

Bit 28 **SD0PID:** Set DATA0 PID

Applies to interrupt/bulk OUT endpoints only.

Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.

Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake

Applies to non-control, non-isochronous OUT endpoints only (access type is rw).

The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Applies to control endpoints only (access type is rs).

The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

This is the transfer type supported by this logical endpoint.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

Bit 17 **NAKSTS:** NAK status

Indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

The core stops receiving any data on an OUT endpoint, even if there is space in the Rx FIFO to accommodate the incoming packet.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **DPID:** Endpoint data PID

Applies to interrupt/bulk OUT endpoints only.

Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.

0: DATA0

1: DATA1

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

50.15.59 OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx)

Valid for ISO endpoints, see previous section for INT/BULK endpoints.

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0xB00 + 0x20 * x, (x = 1 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SODD FRM	SEVN FRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP[1:0]		NAK STS	EO NUM
rs	rs	w	w	w	w					rw	rw	rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable

Applies to IN and OUT endpoints.

The application sets this bit to start transmitting data on an endpoint.

The core clears this bit before setting any of the following interrupts on this endpoint:

– SETUP phase done

– Endpoint disabled

– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SODDFRM:** Set odd frame

Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 **SEVNFRM:** Set even frame

Applies to isochronous OUT endpoints only.

Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.

Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake

Applies to non-control, non-isochronous OUT endpoints only (access type is rw).

The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Applies to control endpoints only (access type is rs).

The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

This is the transfer type supported by this logical endpoint.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

Bit 17 **NAKSTS:** NAK status

Indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

The core stops receiving any data on an OUT endpoint, even if there is space in the Rx FIFO to accommodate the incoming packet.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM:** Even/odd frame

Applies to isochronous IN and OUT endpoints only.

Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.

0: Even frame

1: Odd frame

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

50.15.60 OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx)

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using endpoint enable bit of the OTG_DOEPCTLx registers (EPENA bit in OTG_DOEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Address offset: 0xB10 + 0x20 * x, (x = 1 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RXDPID[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XFRSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **RXDPID[1:0]:**

Condition: isochronous OUT endpoints
Received data PID
This is the data PID received in the last packet for this endpoint.
00: DATA0
01: DATA2
10: DATA1
11: MDATA
Condition: control OUT endpoints
STUPCNT[1:0]: SETUP packet count
This field specifies the number of back-to-back SETUP data packets the endpoint can receive.
01: 1 packet
10: 2 packets
11: 3 packets

Bits 28:19 **PKTCNT[9:0]:** Packet count

Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.
This field is decremented every time a packet (maximum size or short packet) is written to the Rx FIFO.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size

This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.

50.15.61 OTG power and clock gating control register (OTG_PCGCCTL)

This register is available in host and device modes.

Address offset: 0xE00

Reset value: 0x200B 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	PHY SLEEP	ENL1 GTG	PHY SUSP	Res.	Res.	GATE HCLK	STPP CLK
								r	r	rw	r			rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **SUSP:** Deep Sleep

This bit indicates that the PHY is in Deep Sleep when in L1 state.

Bit 6 **PHYSLEEP:** PHY in Sleep

This bit indicates that the PHY is in the Sleep state.

Bit 5 **ENL1GTG:** Enable sleep clock gating

When this bit is set, core internal clock gating is enabled in Sleep state if the core cannot assert utmi_l1_suspend_n. When this bit is not set, the PHY clock is not gated in Sleep state.

Bit 4 **PHYSUSP:** PHY suspended

Indicates that the PHY has been suspended. This bit is updated once the PHY is suspended after the application has set the STPPCLK bit.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **GATEHCLK:** Gate HCLK

The application sets this bit to gate HCLK to modules other than the AHB Slave and Master and wakeup logic when the USB is suspended or the session is not valid. The application clears this bit when the USB is resumed or a new session starts.

Bit 0 **STPPCLK:** Stop PHY clock

The application sets this bit to stop the PHY clock when the USB is suspended, the session is not valid, or the device is disconnected. The application clears this bit when the USB is resumed or a new session starts.

50.15.62 OTG power and clock gating control register 1 (OTG_PCGCCTL1)

This register is available in host and device modes.

Address offset: 0xE04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAM GATE EN	CNTGATECLK [1:0]		GATE EN
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RAMGATEEN:** Enable RAM clock gating

Enable gating of the FIFO RAM.

Bits 2:1 **CNTGATECLK[1:0]:** Counter for clock gating

Indicates to the controller how many PHY clock cycles and AHB clock cycles of 'IDLE' (no activity) the controller waits for before gating the respective PHY and AHB clocks internal to the controller.

00: 64 clocks

01: 128 clocks

10: Reserved

11: Reserved

Bit 0 **GATEEN:** Enable active clock gating

The application programs GATEEN to enable active clock gating feature for the PHY and AHB clocks.

50.15.63 OTG_FS register map

The table below gives the USB OTG register map and reset values.

Table 591. OTG_FS register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	OTG_ GOTGCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CURMOD	OTGVER	BSVLD	ASVLD	DBCT	CIDSTS			Res.	EHEN					BVALOVAL	BVALOEN	AVALOVAL	AVALOEN	VBVALOVA	VBVALOEN		
	Reset value											0	0	X	0	0	1				0					X	X	0	0	0	0		
0x004	OTG_ GOTGINT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADTOCHG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEDET	Res.	Res.
	Reset value														0								-								0		
0x008	OTG_ GAHBCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTXFELVL	TXFELVL	Res.	Res.	Res.	Res.	Res.	Res.	GINTMSK
	Reset value																								0	0							0
0x00C	OTG_ GUSBCFG	Res.	FDMOD	FHMOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRDT				Res.	Res.	Res.	PHYSEL	Res.	Res.	Res.	TOCAL			
	Reset value		X	0																0	1	0	1				1				0	0	0
0x010	OTG_ GRSTCTL	AHBIDL	Res.	CSRSTDONE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFNUM					TXFFLSH	RXFFLSH	Res.	FCRST	PSRST	CSRST
	Reset value	1		0																			0	0	0	0	0	0	0		0	0	0
0x014	OTG_ GINTSTS	WKUPINT	SRQINT	DISCINT	CIDSCHG	LPMINT	PTXFE	HCINT	HPRTINT	RSTDET	Res.	IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Res.	Res.	EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Res.	Res.	GONAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
	Reset value	0	X	0	X	0	1	0	0	0		0	0	0	0			0	0	0	0	0	0			0	0	1	0	0	0	0	0
0x018	OTG_ GINTMSK (Host mode)	WUIM	SRQIM	DISCINT	CIDSCHGM	LPMINTM	PTXFEM	HCIM	PRTIM	Res.	Res.	IPXFRM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Res.
	Reset value	0	0	0	0	0	0	0	0			0																0	0	0	0	0	
	OTG_ GINTMSK (Device mode)	WUIM	SRQIM	Res.	CIDSCHGM	LPMINTM	Res.	Res.	Res.	RSTDETM	Res.	IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	Res.	Res.	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Res.	Res.	GONAKEFFM	GINAKEFFM	Res.	RXFLVLM	SOFM	OTGINT	MMISM	Res.
	Reset value	0	0		0	0				0		0	0	0	0			0	0	0	0	0	0			0	0		0	0	0	0	

Table 591. OTG_FS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	**OTG_ GRXSTSR (Host mode)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS				DPID		BCNT											CHNUM			
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	OTG_ GRXSTSR (Device mode)	Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM				PKTSTS				DPID		BCNT											EPNUM			
	Reset value					0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**OTG_ GRXSTSP (Host mode)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS				DPID		BCNT											CHNUM			
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	OTG_ GRXSTSP (Device mode)	Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM				PKTSTS				DPID		BCNT											EPNUM			
	Reset value					0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	**OTG_ GRXFSIZ**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFD															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x028	**OTG_ HNPTXFSIZ/ OTG_ DIEPTXF0**	NPTXFD/TX0FD																NPTXFSA/TX0FSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x02C	**OTG_ HNPTXSTS**	Res.	NPTXQTOP							NPTQXSAV								NPTXFSAV															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x038	**OTG_ GCCFG**	Res.	Res.	Res.	IDPULLUPEN	Res.	Res.	FORCEHOSTPD	Res.	Res.	Res.	VBDEN	SDEN	PDEN	Res.	BCDEN	PWRDWN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SESSVLD	PS2DET	SDET	PDET	Res.
	Reset value				0			0				0	0	0		0	0												X	X	X	X	
0x03C	**OTG_CID**	PRODUCT_ID																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	**OTG_ GLPMCFG**	Res.	Res.	Res.	ENBESL	LPMR CNTSTS			SNDLPM	LPM RCNT			LPMCHIDX				L1RSMOK	SLPSTS	LPM RSP		L1DSEN	BESLTHRS			L1SSEN	REMWAKE	BESL				LPMACK	LPMEN	
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	**OTG_ HPTXFSIZ**	PTXFSIZ																PTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x104	**OTG_ DIEPTXF1**	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Table 591. OTG_FS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
. . . .	. . . .	. . . .																															
0x114	**OTG_ DIEPTXF5**	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
0x400	**OTG_ HCFG**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSLSS	FSLS PCS	
	Reset value																														0	0	0
0x404	**OTG_ HFIR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLDCTRL	FRIVL															
	Reset value																0	1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
0x408	**OTG_ HFNUM**	FTREM																FRNUM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x410	**OTG_ HPTXSTS**	PTXQTOP								PTXQSAV								PTXFSAVL															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x414	**OTG_ HAINT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HAINT															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x418	**OTG_ HAINTMSK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HAINTM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x440	**OTG_ HPRT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSPD	PTCTL		PPWR	PLSTS	Res.	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS				
	Reset value														0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0
0x500	**OTG_ HCCHAR0**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP	LSDEV	Res.	EPDIR	EPNUM					MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x508	**OTG_ HCINT0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x508	**OTG_ HCINT0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0

Table 591. OTG_FS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x50C	**OTG_ HCINTMSK0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM		ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
	Reset value																						0	0	0	0		0	0	0		0	0
0x510	**OTG_ HCTSIZ0**	DOPNG	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	Iterating preceding block of registers starting at offset 0x500.																															
0x660	**OTG_ HCCHAR11**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP	LSDEV	Res	EPDIR	EPNUM					MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x668	**OTG_ HCINT11**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x66C	**OTG_ HCINTMSK11**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM		ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
	Reset value																						0	0	0	0		0	0	0		0	0
0x670	**OTG_ HCTSIZ11**	DOPNG	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x800	**OTG_ DCFG**	Res	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ERRATIM	XCVRDLY	Res	PFIVL		DAD							Res.	NZLSOHSK	DSPD	
	Reset value																	0	0		0	0	0	0	0	0	0	0	0		0	0	0
0x804	**OTG_ DCTL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSBESLRJCT	Res.	Res.	Res.	Res.	Res.	Res.	POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
	Reset value														0							0	0	0	0	0	0	0	0	0	0	1	0
0x808	**OTG_ DSTS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEV LN STS		FNSOF											Res.	Res.	Res.	Res.				EERR	ENUMSPD		SUSPSTS
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0

Table 591. OTG_FS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x810	**OTG_ DIEPMSK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAKM	Res.	Res.	Res.	Res.	Res.	Res.	INEPNEM	INEPNMM	ITTXFEMSK	TOM	Res.	EPDM	XFRCM
	Reset value																			0							0	0	0	0		0	0
0x814	**OTG_ DOEPMSK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAKMSK	BERRM	Res.	Res.	Res.	OUTPKTERRM	Res.	Res.	Res.	OTEPDM	STUPM	Res.	EPDM	XFRCM
	Reset value																			0	0				0				0	0		0	0
0x818	**OTG_ DAINT**	OEPINT																IEPINT															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x81C	**OTG_ DAINTMSK**	OEPM																IEPM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x834	**OTG_DIE PEMPMSK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTXFEM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x838	**OTG_ DEACHINT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OEP1INT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IEP1INT	Res.
	Reset value															0																0	
0x83C	**OTG_DEACHI NTMSK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OEP1INTM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IEP1INTM	Res.
	Reset value															0																0	
0x844	**OTG_ DIEPEACHM**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAKM	Res.	Res.	Res.	Res.	TXFURM	Res.	INEPNEM	Res.	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM
	Reset value																			0					0	-	0		0	0	0	0	0
0x884	**OTG_ DOEPEACH MSK1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NYETMSK	NAKMSK	BERRM	Res.	Res.	Res.	Res.	OUTPKTERRM	Res.	Res.	Res.	OTEPDM	STUPM	AHBERRM	EPDM	XFRCM
	Reset value																	0	0	0					0				0	0	0	0	0
0x900	**OTG_ DIEPCTL0**	EPENA	EPDIS	Res.	Res.	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	Res.	USBAEP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ	
	Reset value	0	0			0	0	0	0	0	0	0		0	0	0		1														0	0

Table 591. OTG_FS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x908	**OTG_ DIEPINT0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	Res.	Res.	TXFE	INEPNE	INEPNM	ITTXFE	TOC	Res.	EPDISD	XFRC
	Reset value																			0		0				1	0	0	0	0		0	0
0x910	**OTG_ DIEPTSIZ0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKT CNT		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ						
	Reset value												0	0													0	0	0	0	0	0	0
0x918	**OTG_ DTXFSTS0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x920	**OTG_ DIEPCTL1 (INT/BULK)**	EPENA	EPDIS	SD1PID	SD0PID	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
	OTG_ DIEPCTL1 (ISO)	EPENA	EPDIS	SODDFRM	SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	EONUM	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x928	**OTG_ DIEPINT1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	Res.	Res.	TXFE	INEPNE	INEPNM	ITTXFE	TOC	Res.	EPDISD	XFRC
	Reset value																			0		0				1	0	0	0	0		0	0
0x930	**OTG_ DIEPTSIZ1**	Res.	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x938	**OTG_ DTXFSTS1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
. . . .	. . . .	. Iterating preceding block of registers starting at offset 0x920. .																															
0x9A0	**OTG_ DIEPCTL5 (INT/BULK)**	EPENA	EPDIS	SD1PID	SD0PID	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
	OTG_ DIEPCTL5 (ISO)	EPENA	EPDIS	SODDFRM	SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	EONUM	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0

Table 591. OTG_FS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x9A8	**OTG_ DIEPINT5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	Res.	Res.	TXFE	INEPNE	INEPNM	ITTXFE	TOC	Res.	EPDISD	XFRC
	Reset value																			0		0				1	0	0	0	0		0	0
0x9B8	**OTG_ DTXFSTS5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x9B0	**OTG_ DIEPTSIZ5**	Res.	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB00	**OTG_ DOEPCTL0**	EPENA	EPDIS	Res.	Res.	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP		NAKSTS	Res.	USBAEP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ	
	Reset value	0	0			0	0					0	0	0	0	0		1														0	0
0xB08	**OTG_ DOEPINT0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	BERR	Res.	Res.	Res.	Res.	Res.	Res.	STSPHSRX	OTEPDIS	STUP	Res.	EPDISD	XFRC
	Reset value																			0	0							0	0	0		0	0
0xB10	**OTG_ DOEPTSIZ0**	Res.	STUP CNT		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ					
	Reset value		0	0										0													0	0	0	0	0	0	0
0xB20	**OTG_ DOEPCTL1 (INT/BULK)**	EPENA	EPDIS	SD1PID	SD0PID	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EP TYP		NAKSTS	DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
	OTG_ DOEPCTL1 (ISO)	EPENA	EPDIS	SODDFRM	SEVNFRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EP TYP		NAKSTS	EONUM	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB28	**OTG_ DOEPINT1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	BERR	Res.	Res.	Res.	Res.	Res.	Res.	STSPHSRX	OTEPDIS	STUP	Res.	EPDISD	XFRC
	Reset value																			0	0							0	0	0		0	0
0xB30	**OTG_ DOEPTSIZ1**	Res.	RXDPID/ STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	Iterating preceding block of registers starting at offset 0xB20.																															

Table 591. OTG_FS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xBA0	OTG_ DOEPCTL5 (INT/BULK)	EPENA	EPDIS	SD1PID	SD0PID	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EP TYP		NAKSTS	DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
	OTG_ DOEPCTL5 (ISO)	EPENA	EPDIS	SODDFRM	SEVNFRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EP TYP		NAKSTS	EONUM	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xBA8	OTG_ DOEPINT5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	BERR	Res.	Res.	Res.	Res.	Res.	Res.	STSPHSRX	OTEPDIS	STUP	Res.	EPDISD	XFRC
	Reset value																			0	0					-		0	0	0		0	0
0xBB0	OTG_ DOEPTSIZ5	Res.	RXDPID/ STUPCNT		PKTCNT											XFRSIZ																	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE00	OTG_ PCGCCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	PHYSLEEP	ENL1GTG	PHYSUSP	Res.	Res.	GATEHCLK	STPPCLK
	Reset value																									0	0	0	0			0	0
0xE04	OTG_ PCGCCTL1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAMGATEEN	CNTGATECLK		GATEEN
	Reset value																													0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

50.16 OTG_FS programming model

50.16.1 Core initialization

The application must perform the core initialization sequence. If the cable is connected during power-up, the current mode of operation bit in the OTG_GINTSTS (CMOD bit in OTG_GINTSTS) reflects the mode. The OTG_FS controller enters host mode when an “A” plug is connected or device mode when a “B” plug is connected.

This section explains the initialization of the OTG_FS controller after power-on. The application must follow the initialization sequence irrespective of host or device mode operation. All core global registers are initialized according to the core’s configuration:

1. Program the following fields in the OTG_GAHBCFG register:
   - Global interrupt mask bit GINTMSK = 1
   - Rx FIFO non-empty (RXFLVL bit in OTG_GINTSTS)
   - Periodic Tx FIFO empty level
2. Program the following fields in the OTG_GUSBCFG register:
   - OTG_FS timeout calibration field
   - USB turnaround time field
3. The software must unmask the following bits in the OTG_GINTMSK register:

   OTG interrupt mask

   Mode mismatch interrupt mask
4. The software can read the CMOD bit in OTG_GINTSTS to determine whether the OTG_FS controller is operating in host or device mode.

50.16.2 Host initialization

To initialize the core as host, the application must perform the following steps:

1. Program the HPRTINT in the OTG_GINTMSK register to unmask
2. Program the OTG_HCFG register to select full-speed host
3. Program the PPWR bit in OTG_HPRT to 1. This drives V_{BUS} on the USB.
4. Wait for the PCDET interrupt in OTG_HPRT0. This indicates that a device is connecting to the port.
5. Program the PRST bit in OTG_HPRT to 1. This starts the reset process.
6. Wait at least 10 ms for the reset process to complete.
7. Program the PRST bit in OTG_HPRT to 0.
8. Wait for the PENCHNG interrupt in OTG_HPRT.
9. Read the PSPD bit in OTG_HPRT to get the enumerated speed.
10. Program the HFIR register with a value corresponding to the selected PHY clock 1
11. Program the FSLSPCS field in the OTG_HCFG register following the speed of the device detected in step 9. If FSLSPCS has been changed a port reset must be performed.
12. Program the OTG_GRXFSIZ register to select the size of the receive FIFO.
13. Program the OTG_HNPTXFSIZ register to select the size and the start address of the Non-periodic transmit FIFO for non-periodic transactions.
14. Program the OTG_HPTXFSIZ register to select the size and start address of the periodic transmit FIFO for periodic transactions.

To communicate with devices, the system software must initialize and enable at least one channel.

50.16.3 Device initialization

The application must perform the following steps to initialize the core as a device on power-up or after a mode change from host to device.

1. Program the following fields in the OTG_DCFG register:
   - Device speed
   - Non-zero-length status OUT handshake
   - Periodic Frame Interval
2. Clear the DCTL.SDIS bit. The core issues a connect after this bit is cleared.
3. Program the OTG_GINTMSK register to unmask the following interrupts:
   - USB reset
   - Enumeration done
   - Early suspend
   - USB suspend
   - SOF
4. Wait for the USBRST interrupt in OTG_GINTSTS. It indicates that a reset has been detected on the USB that lasts for about 10 ms on receiving this interrupt.
5. Wait for the ENUMDNE interrupt in OTG_GINTSTS. This interrupt indicates the end of reset on the USB. On receiving this interrupt, the application must read the OTG_DSTS

register to determine the enumeration speed and perform the steps listed in *Endpoint initialization on enumeration completion on page 3151*.

At this point, the device is ready to accept SOF packets and perform control transfers on control endpoint 0.

50.16.4 Host programming model

Channel initialization

The application must initialize one or more channels before it can communicate with connected devices. To initialize and enable a channel, the application must perform the following steps:

1. Program the OTG_GINTMSK register to unmask the following:
2. Channel interrupt
   - Non-periodic transmit FIFO empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
   - Non-periodic transmit FIFO half-empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
3. Program the OTG_HAINTMSK register to unmask the selected channels' interrupts.
4. Program the OTG_HCINTMSK register to unmask the transaction-related interrupts of interest given in the host channel interrupt register.
5. Program the selected channel's OTG_HCTSIZx register with the total transfer size, in bytes, and the expected number of packets, including short packets. The application must program the PID field with the initial data PID (to be used on the first OUT transaction or to be expected from the first IN transaction).
6. Program the OTG_HCCHARx register of the selected channel with the device's endpoint characteristics, such as type, speed, direction, and so forth. (The channel can be enabled by setting the channel enable bit to 1 only when the application is ready to transmit or receive any packet).

Halting a channel

The application can disable any channel by programming the OTG_HCCHARx register with the CHDIS and CHENA bits set to 1. This enables the OTG_FS host to flush the posted requests (if any) and generates a channel halted interrupt. The application must wait for the CHH interrupt in OTG_HCINTx before reallocating the channel for other transactions. The OTG_FS host does not interrupt the transaction that has already been started on the USB.

Before disabling a channel, the application must ensure that there is at least one free space available in the non-periodic request queue (when disabling a non-periodic channel) or the periodic request queue (when disabling a periodic channel). The application can simply flush the posted requests when the request queue is full (before disabling the channel), by programming the OTG_HCCHARx register with the CHDIS bit set to 1 which automatically clears the CHENA bit to 0.

The application is expected to disable a channel on any of the following conditions:

1. When an STALL, TXERR, BBERR or DTERR interrupt in OTG_HCINTx is received for an IN or OUT channel. The application must be able to receive other interrupts (DTERR, Nak, data, TXERR) for the same channel before receiving the halt.
2. When a DISCINT (disconnect device) interrupt in OTG_GINTSTS is received, the application must check the PCSTS bit in OTG_HPRT. When the device directly connected to the host is disconnected, PCSTS is reset. The application must issue a soft reset using CSRST in OTG_GRSTCTL to ensure that all channels are cleared. Refer to the bit description for the full implications of this action. When the device is reconnected, the host must issue a USB reset.
3. When the application aborts a transfer before normal completion.

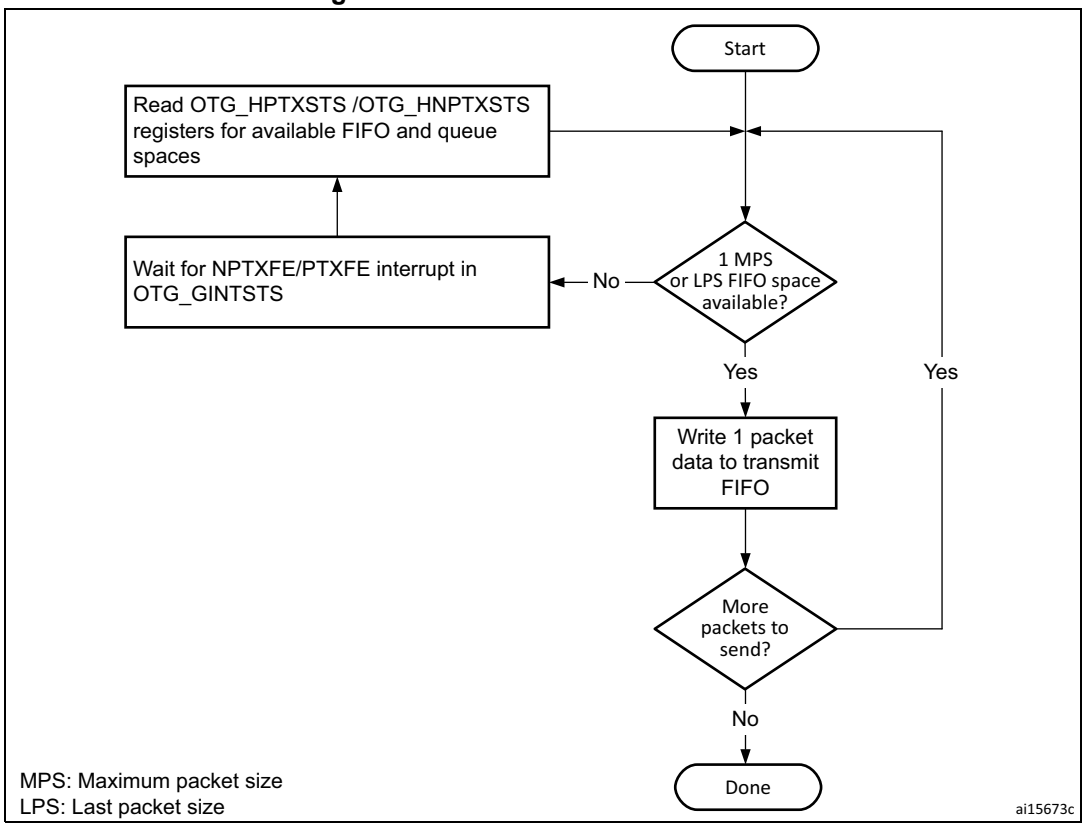
Operational model

The application must initialize a channel before communicating to the connected device. This section explains the sequence of operation to be performed for different types of USB transactions.

- **Writing the transmit FIFO**

  The OTG_FS host automatically writes an entry (OUT request) to the periodic/non-periodic request queue, along with the last 32-bit word write of a packet. The application must ensure that at least one free space is available in the periodic/non-periodic request queue before starting to write to the transmit FIFO. The application must always write to the transmit FIFO in 32-bit words. If the packet size is non-32-bit word aligned, the application must use padding. The OTG_FS host determines the actual packet size based on the programmed maximum packet size and transfer size.

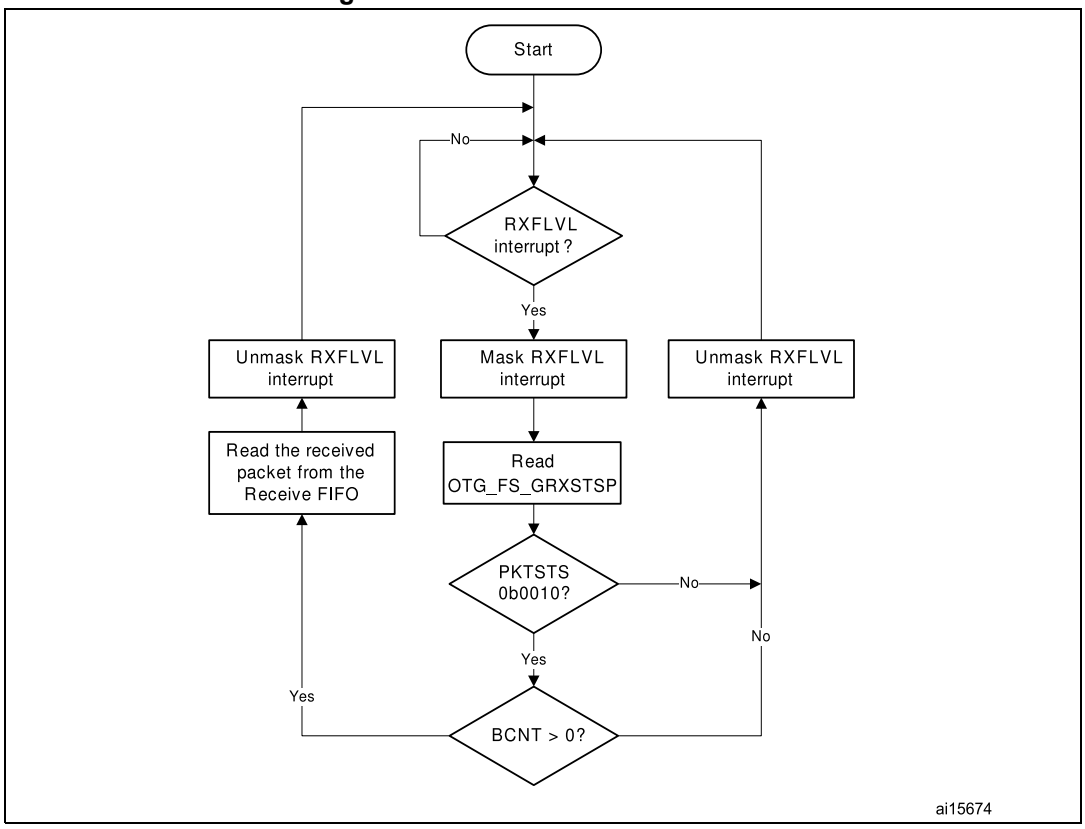
Figure 1019. Transmit FIFO write task



- **Reading the receive FIFO**

  The application must ignore all packet statuses other than IN data packet (bx0010).

Figure 1020. Receive FIFO read task



- **Bulk and control OUT/SETUP transactions**

  A typical bulk or control OUT/SETUP pipelined transaction-level operation is shown in *Figure 1021*. See channel 1 (ch_1). Two bulk OUT packets are transmitted. A control SETUP transaction operates in the same way but has only one packet. The assumptions are:

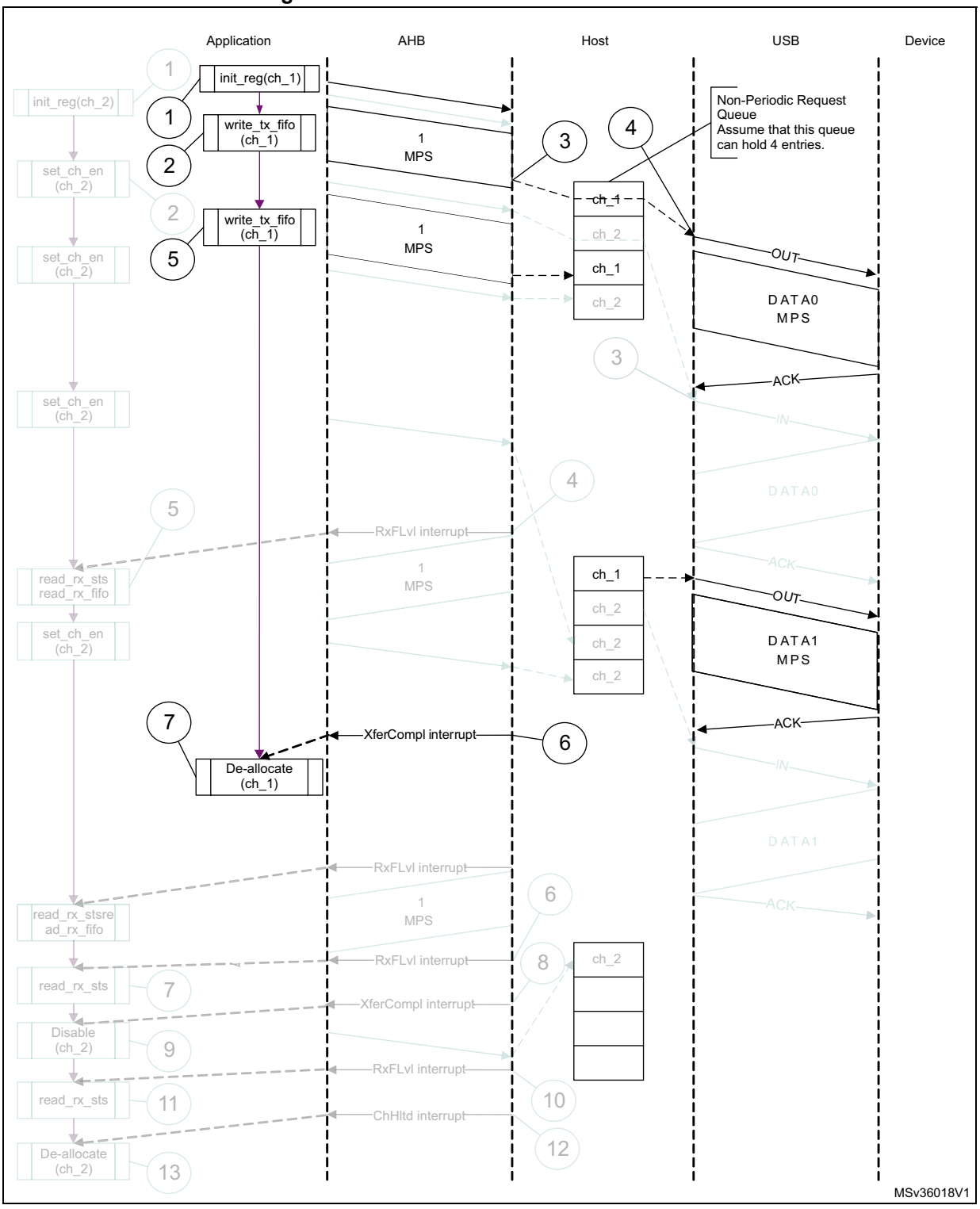
  – The application is attempting to send two maximum-packet-size packets (transfer size = 1, 024 bytes).
  – The non-periodic transmit FIFO can hold two packets (128 bytes for FS).
  – The non-periodic request queue depth = 4.

- **Normal bulk and control OUT/SETUP operations**

  The sequence of operations in (channel 1) is as follows:

1. Initialize channel 1
2. Write the first packet for channel 1
3. Along with the last word write, the core writes an entry to the non-periodic request queue
4. As soon as the non-periodic queue becomes non-empty, the core attempts to send an OUT token in the current frame
5. Write the second (last) packet for channel 1
6. The core generates the XFRC interrupt as soon as the last transaction is completed successfully
7. In response to the XFRC interrupt, de-allocate the channel for other transfers
8. Handling non-ACK responses

Figure 1021. Normal bulk/control OUT/SETUP



1. The grayed elements are not relevant in the context of this figure.

The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions is shown in the following code samples.

- **Interrupt service routine for bulk/control OUT/SETUP and bulk/control IN transactions**

  a) Bulk/control OUT/SETUP

```
Unmask (NAK/TXERR/STALL/XFRC)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   De-allocate Channel
   }
else if (STALL)
   {
   Transfer Done = 1
   Unmask CHH
   Disable Channel
   }
else if (NAK or TXERR )
   {
   Rewind Buffer Pointers
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   else
      {
      Reset Error Count
      }
      }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
```

```
   else if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

The application is expected to write the data packets into the transmit FIFO when the space is available in the transmit FIFO and the request queue. The application can make use of the NPTXFE interrupt in OTG_GINTSTS to find the transmit FIFO space.

b) Bulk/control IN

```
Unmask (TXERR/XFRC/BBERR/STALL/DTERR)
if (XFRC)
   {
   Reset Error Count
   Unmask CHH
   Disable Channel
   Reset Error Count
   Mask ACK
   }
else if (TXERR or BBERR or STALL)
   {
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
else if (ACK)
   {
   Reset Error Count
   Mask ACK
   }
```

```
else if (DTERR)
   {
   Reset Error Count
   }
```

The application is expected to write the requests as and when the request queue space is available and until the XFRC interrupt is received.

- **Bulk and control IN transactions**

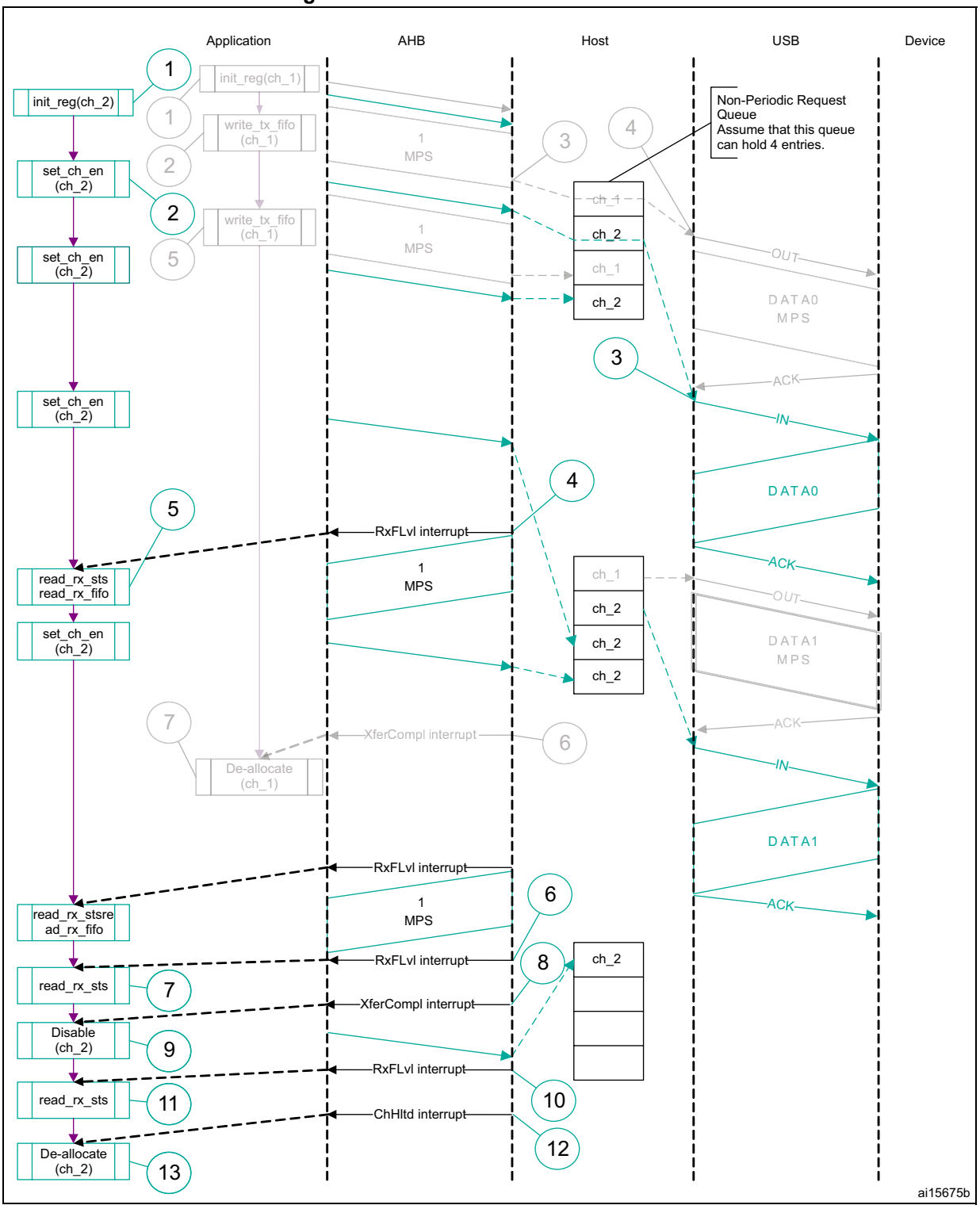
  A typical bulk or control IN pipelined transaction-level operation is shown in *Figure 1022*. See channel 2 (ch_2). The assumptions are:

  – The application is attempting to receive two maximum-packet-size packets (transfer size = 1 024 bytes).

  – The receive FIFO can contain at least one maximum-packet-size packet and two status words per packet (72 bytes for FS).

  – The non-periodic request queue depth = 4.

Figure 1022. Bulk/control IN transactions



1. The grayed elements are not relevant in the context of this figure.

The sequence of operations is as follows:

1. Initialize channel 2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the non-periodic request queue.
3. The core attempts to send an IN token after completing the current OUT transaction.
4. The core generates an RXFLVL interrupt as soon as the received packet is written to the receive FIFO.
5. In response to the RXFLVL interrupt, mask the RXFLVL interrupt and read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. Following this, unmask the RXFLVL interrupt.
6. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO.
7. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in OTG_GRXSTSR ≠ 0b0010).
8. The core generates the XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, disable the channel and stop writing the OTG_HCCHAR2 register for further requests. The core writes a channel disable request to the non-periodic request queue as soon as the OTG_HCCHAR2 register is written.
10. The core generates the RXFLVL interrupt as soon as the halt status is written to the receive FIFO.
11. Read and ignore the receive packet status.
12. The core generates a CHH interrupt as soon as the halt status is popped from the receive FIFO.
13. In response to the CHH interrupt, de-allocate the channel for other transfers.
14. Handling non-ACK responses

- **Control transactions**

  Setup, data, and status stages of a control transfer must be performed as three separate transfers. setup-, data- or status-stage OUT transactions are performed similarly to the bulk OUT transactions explained previously. Data- or status-stage IN transactions are performed similarly to the bulk IN transactions explained previously. For all three stages, the application is expected to set the EPTYP field in

OTG_HCCHAR1 to control. During the setup stage, the application is expected to set the PID field in OTG_HCTSIZ1 to SETUP.

- **Interrupt OUT transactions**

  A typical interrupt OUT operation is shown in *Figure 1023*. The assumptions are:

  – The application is attempting to send one packet in every frame (up to 1 maximum packet size), starting with the odd frame (transfer size = 1 024 bytes)

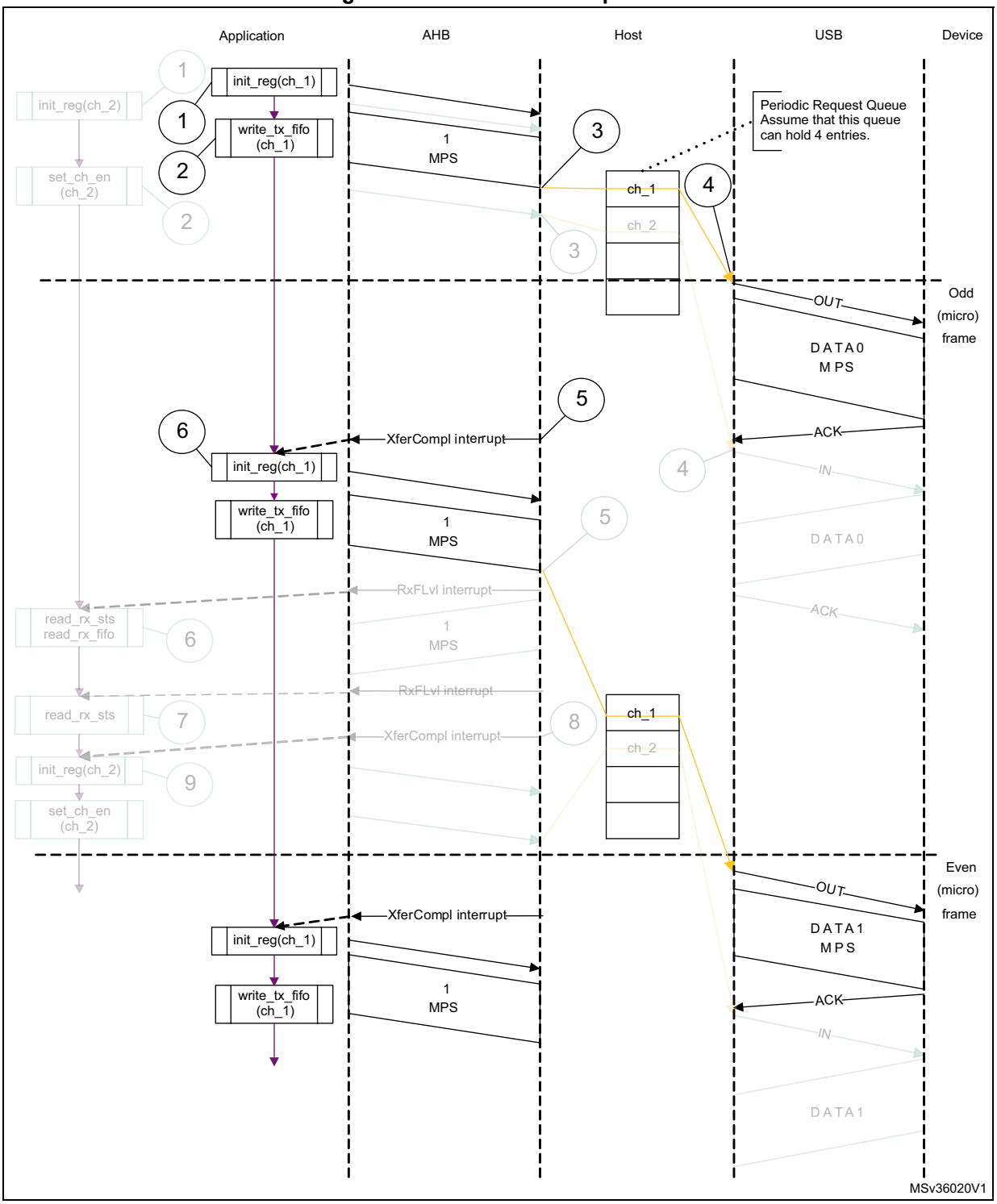
  – The periodic transmit FIFO can hold one packet (1 Kbyte)

  – Periodic request queue depth = 4

  The sequence of operations is as follows:

1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG_FS host writes an entry to the periodic request queue.
4. The OTG_FS host attempts to send an OUT token in the next (odd) frame.
5. The OTG_FS host generates an XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.

Figure 1023. Normal interrupt OUT



1. The grayed elements are not relevant in the context of this figure.

- **Interrupt service routine for interrupt OUT/IN transactions**

  a) Interrupt OUT

`Unmask (NAK/TXERR/STALL/XFRC/FRMOR)`

```
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   De-allocate Channel
   }
else
   if (STALL or FRMOR)
      {
      Mask ACK
      Unmask CHH
      Disable Channel
      if (STALL)
         {
         Transfer Done = 1
         }
      }
   else
      if (NAK or TXERR)
         {
         Rewind Buffer Pointers
         Reset Error Count
         Mask ACK
         Unmask CHH
         Disable Channel
         }
      else
         if (CHH)
         {
         Mask CHH
         if (Transfer Done or (Error_count == 3))
            {
            De-allocate Channel
            }
         else
            {
            Re-initialize Channel (in next b_interval - 1 Frame)
            }
         }
      else
         if (ACK)
         {
         Reset Error Count
         Mask ACK
         }
```

```
The application uses the NPTXFE interrupt in OTG_GINTSTS to find the
transmit FIFO space.
Interrupt IN
Unmask (NAK/TXERR/XFRC/BBERR/STALL/FRMOR/DTERR)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   if (OTG_HCTSIZx.PKTCNT == 0)
      {
      De-allocate Channel
      }
   else
      {
      Transfer Done = 1
      Unmask CHH
      Disable Channel
      }
   }
else
   if (STALL or FRMOR or NAK or DTERR or BBERR)
      {
      Mask ACK
      Unmask CHH
      Disable Channel
      if (STALL or BBERR)
         {
         Reset Error Count
         Transfer Done = 1
         }
      else
         if (!FRMOR)
         {
         Reset Error Count
         }
      }
else
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      Unmask CHH
      Disable Channel
      }
else
```

```
   if (CHH)
      {
      Mask CHH
      if (Transfer Done or (Error_count == 3))
         {
         De-allocate Channel
         }
      else
         Re-initialize Channel (in next b_interval - 1 /Frame)
      }
   }
else
   if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

- **Interrupt IN transactions**

  The assumptions are:

  – The application is attempting to receive one packet (up to 1 maximum packet size) in every frame, starting with odd (transfer size = 1 024 bytes).

  – The receive FIFO can hold at least one maximum-packet-size packet and two status words per packet (1 031 bytes).

  – Periodic request queue depth = 4.

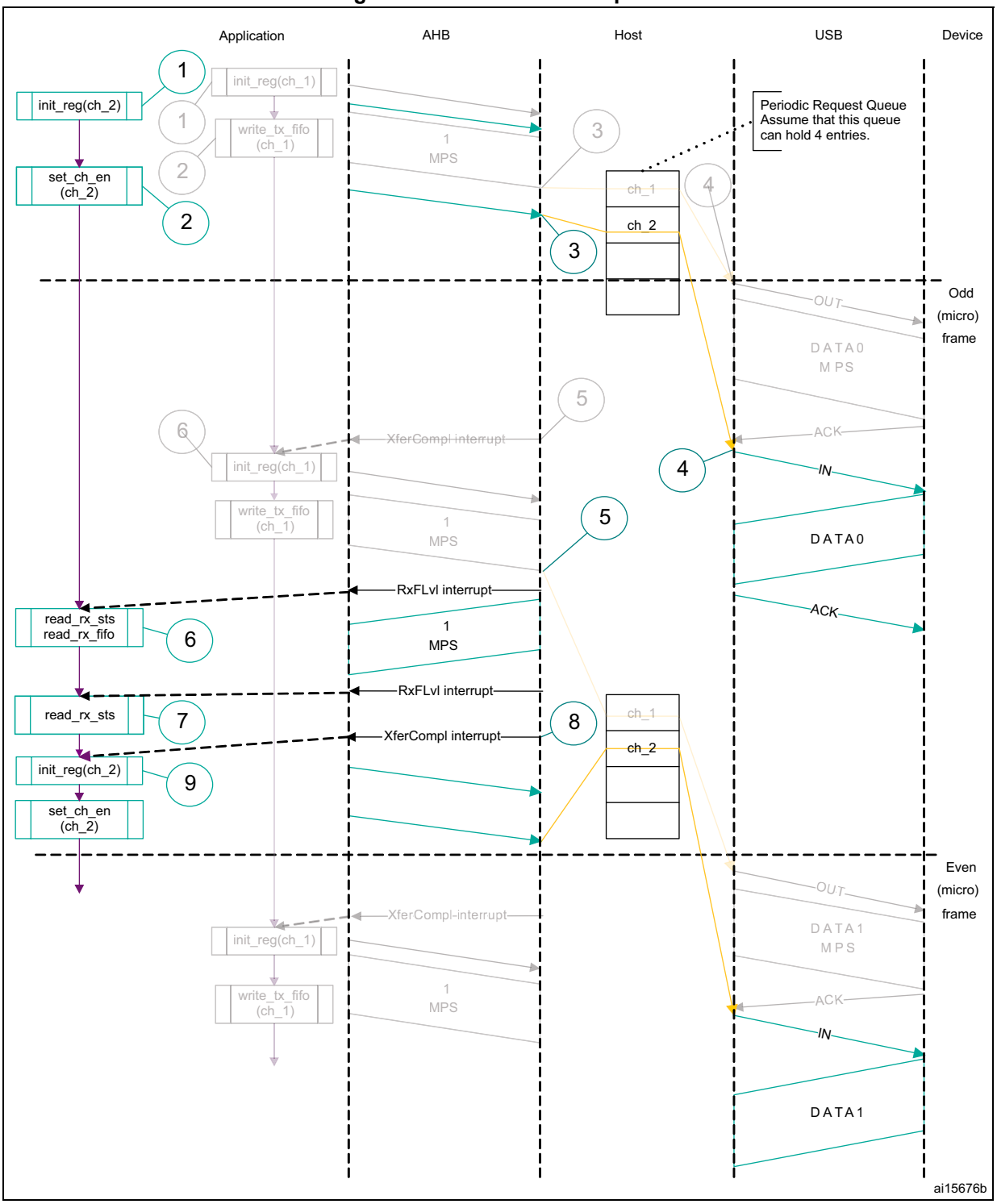
- **Normal interrupt IN operation**

  The sequence of operations is as follows:

1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
3. The OTG_FS host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
4. The OTG_FS host attempts to send an IN token in the next (odd) frame.
5. As soon as the IN packet is received and written to the receive FIFO, the OTG_FS host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask after reading the entire packet.
7. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in GRXSTSR ≠ 0b0010).
8. The core generates an XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If the PKTCNT bit in OTG_HCTSIZ2 is not equal to 0, disable the channel before re-

initializing the channel for the next transfer, if any). If PKTCNT bit in OTG_HCTSIZ2 = 0, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.

Figure 1024. Normal interrupt IN



1. The grayed elements are not relevant in the context of this figure.

- **Isochronous OUT transactions**

  A typical isochronous OUT operation is shown in *Figure 1025*. The assumptions are:

  – The application is attempting to send one packet every frame (up to 1 maximum

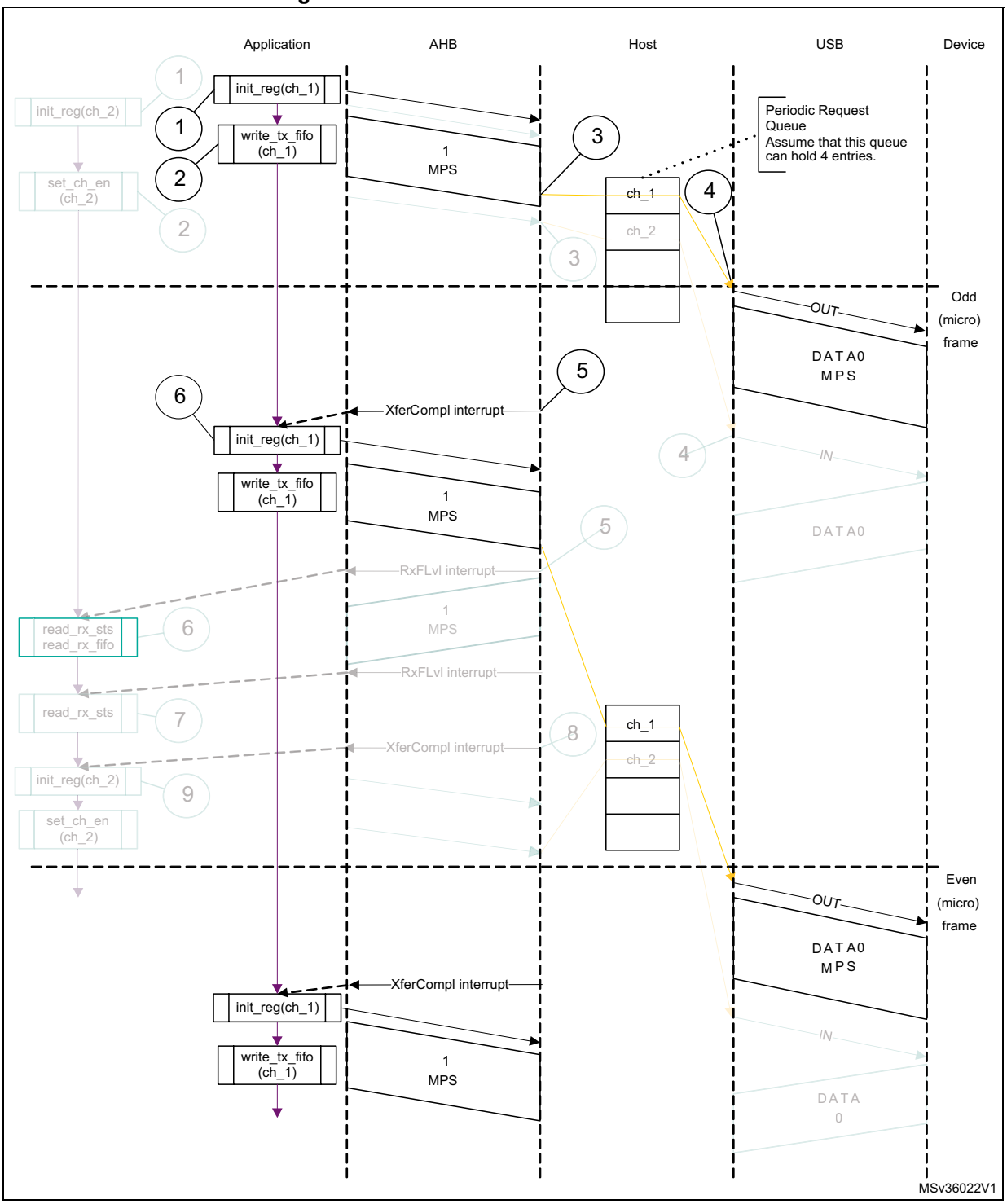
packet size), starting with an odd frame. (transfer size = 1 024 bytes).

– The periodic transmit FIFO can hold one packet (1 Kbyte).
– Periodic request queue depth = 4.

The sequence of operations is as follows:

1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG_FS host writes an entry to the periodic request queue.
4. The OTG_FS host attempts to send the OUT token in the next frame (odd).
5. The OTG_FS host generates the XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.
7. Handling non-ACK responses

Figure 1025. Isochronous OUT transactions



1. The grayed elements are not relevant in the context of this figure.

- **Interrupt service routine for isochronous OUT/IN transactions**

  Code sample: isochronous OUT

```
Unmask (FRMOR/XFRC)
if (XFRC)
```

```
   {
   De-allocate Channel
   }
else
   if (FRMOR)
      {
      Unmask CHH
      Disable Channel
      }
   else
   if (CHH)
      {
      Mask CHH
      De-allocate Channel
      }
Code sample: Isochronous IN
Unmask (TXERR/XFRC/FRMOR/BBERR)
if (XFRC or FRMOR)
   {
   if (XFRC and (OTG_HCTSIZx.PKTCNT == 0))
      {
      Reset Error Count
      De-allocate Channel
      }
   else
      {
      Unmask CHH
      Disable Channel
      }
   }
   else
      if (TXERR or BBERR)
         {
         Increment Error Count
         Unmask CHH
         Disable Channel
         }
      else
         if (CHH)
            {
            Mask CHH
            if (Transfer Done or (Error_count == 3))
               {
               De-allocate Channel
               }
```

```
else
   {
   Re-initialize Channel
   }
}
```

- **Isochronous IN transactions**

  The assumptions are:

  – The application is attempting to receive one packet (up to 1 maximum packet size) in every frame starting with the next odd frame (transfer size = 1 024 bytes).

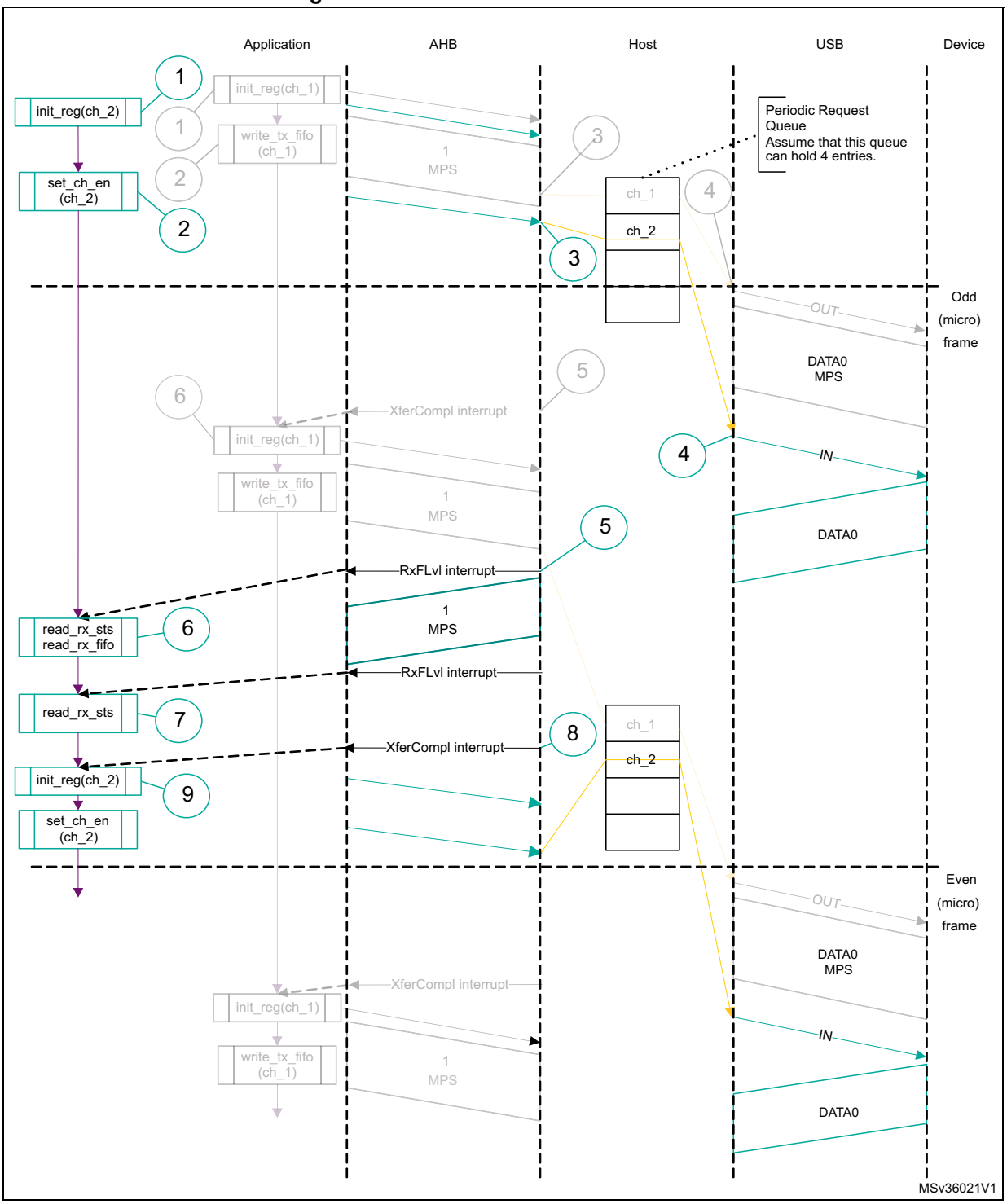
  – The receive FIFO can hold at least one maximum-packet-size packet and two status word per packet (1 031 bytes).

  – Periodic request queue depth = 4.

  The sequence of operations is as follows:

1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
3. The OTG_FS host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
4. The OTG_FS host attempts to send an IN token in the next odd frame.
5. As soon as the IN packet is received and written to the receive FIFO, the OTG_FS host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask it after reading the entire packet.
7. The core generates an RXFLVL interrupt for the transfer completion status entry in the receive FIFO. This time, the application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS bit in OTG_GRXSTSR ≠ 0b0010).
8. The core generates an XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If PKTCNT ≠ 0 in OTG_HCTSIZ2, disable the channel before re-initializing the channel for the next transfer, if any. If PKTCNT = 0 in OTG_HCTSIZ2, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.

Figure 1026. Isochronous IN transactions



1. The grayed elements are not relevant in the context of this figure.

- **Selecting the queue depth**

  Choose the periodic and non-periodic request queue depths carefully to match the number of periodic/non-periodic endpoints accessed.

  The non-periodic request queue depth affects the performance of non-periodic

transfers. The deeper the queue (along with sufficient FIFO size), the more often the core is able to pipeline non-periodic transfers. If the queue size is small, the core is able to put in new requests only when the queue space is freed up.

The core's periodic request queue depth is critical to perform periodic transfers as scheduled. Select the periodic queue depth, based on the number of periodic transfers scheduled in a microframe. If the periodic request queue depth is smaller than the periodic transfers scheduled in a microframe, a frame overrun condition occurs.

- **Handling babble conditions**

  OTG_FS controller handles two cases of babble: packet babble and port babble. Packet babble occurs if the device sends more data than the maximum packet size for the channel. Port babble occurs if the core continues to receive data from the device at EOF2 (the end of frame 2, which is very close to SOF).

  When OTG_FS controller detects a packet babble, it stops writing data into the Rx buffer and waits for the end of packet (EOP). When it detects an EOP, it flushes already written data in the Rx buffer and generates a Babble interrupt to the application.

  When OTG_FS controller detects a port babble, it flushes the Rx FIFO and disables the port. The core then generates a port disabled interrupt (HPRTINT in OTG_GINTSTS, PENCHNG in OTG_HPRT). On receiving this interrupt, the application must determine that this is not due to an overcurrent condition (another cause of the port disabled interrupt) by checking POCA in OTG_HPRT, then perform a soft reset. The core does not send any more tokens after it has detected a port babble condition.

50.16.5 Device programming model

Endpoint initialization on USB reset

1. Set the NAK bit for all OUT endpoints
   - SNAK = 1 in OTG_DOEPCTLx (for all OUT endpoints)
2. Unmask the following interrupt bits
   - INEP0 = 1 in OTG_DAINTMSK (control 0 IN endpoint)
   - OUTEP0 = 1 in OTG_DAINTMSK (control 0 OUT endpoint)
   - STUPM = 1 in OTG_DOEPMSK
   - XFRCM = 1 in OTG_DOEPMSK
   - XFRCM = 1 in OTG_DIEPMSK
   - TOM = 1 in OTG_DIEPMSK
3. Set up the data FIFO RAM for each of the FIFOs
   - Program the OTG_GRXFSIZ register, to be able to receive control OUT data and setup data. If thresholding is not enabled, at a minimum, this must be equal to 1 max packet size of control endpoint 0 + 2 words (for the status of the control OUT data packet) + 10 words (for setup packets).
   - Program the OTG_DIEPTXF0 register (depending on the FIFO number chosen) to be able to transmit control IN data. At a minimum, this must be equal to 1 max packet size of control endpoint 0.
4. Program the following fields in the endpoint-specific registers for control OUT endpoint 0 to receive a SETUP packet
   - STUPCNT = 3 in OTG_DOEPTSIZ0 (to receive up to 3 back-to-back SETUP packets)

At this point, all initialization required to receive SETUP packets is done.

Endpoint initialization on enumeration completion

1. On the Enumeration Done interrupt (ENUMDNE in OTG_GINTSTS), read the OTG_DSTS register to determine the enumeration speed.
2. Program the MPSIZ field in OTG_DIEPCTL0 to set the maximum packet size. This step configures control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed.

At this point, the device is ready to receive SOF packets and is configured to perform control transfers on control endpoint 0.

Endpoint initialization on SetAddress command

This section describes what the application must do when it receives a SetAddress command in a SETUP packet.

1. Program the OTG_DCFG register with the device address received in the SetAddress command
2. Program the core to send out a status IN packet

Endpoint initialization on SetConfiguration/SetInterface command

This section describes what the application must do when it receives a SetConfiguration or SetInterface command in a SETUP packet.

1. When a SetConfiguration command is received, the application must program the endpoint registers to configure them with the characteristics of the valid endpoints in the new configuration.
2. When a SetInterface command is received, the application must program the endpoint registers of the endpoints affected by this command.
3. Some endpoints that were active in the prior configuration or alternate setting are not valid in the new configuration or alternate setting. These invalid endpoints must be deactivated.
4. Unmask the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the OTG_DAINTMSK register.
5. Set up the data FIFO RAM for each FIFO.
6. After all required endpoints are configured; the application must program the core to send a status IN packet.

At this point, the device core is configured to receive and transmit any type of data packet.

Endpoint activation

This section describes the steps required to activate a device endpoint or to configure an existing device endpoint to a new type.

1. Program the characteristics of the required endpoint into the following fields of the OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx register (for OUT or bidirectional endpoints).
   – Maximum packet size
   – USB active endpoint = 1
   – Endpoint start data toggle (for interrupt and bulk endpoints)
   – Endpoint type
   – Tx FIFO number
2. Once the endpoint is activated, the core starts decoding the tokens addressed to that endpoint and sends out a valid handshake for each valid token received for the endpoint.

Endpoint deactivation

This section describes the steps required to deactivate an existing endpoint.

1. In the endpoint to be deactivated, clear the USB active endpoint bit in the OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx register (for OUT or bidirectional endpoints).
2. Once the endpoint is deactivated, the core ignores tokens addressed to that endpoint, which results in a timeout on the USB.

Note: The application must meet the following conditions to set up the device core to handle traffic:
NPTXFEM and RXFLVLM in the OTG_GINTMSK register must be cleared.

Operational model

SETUP and OUT data transfers:

This section describes the internal data flow and application-level operations during data OUT transfers and SETUP transactions.

- **Packet read**

This section describes how to read packets (OUT data and SETUP packets) from the receive FIFO.

1. On catching an RXFLVL interrupt (OTG_GINTSTS register), the application must read the receive status pop register (OTG_GRXSTSP).
2. The application can mask the RXFLVL interrupt (in OTG_GINTSTS) by writing to RXFLVLM = 0 (in OTG_GINTMSK), until it has read the packet from the receive FIFO.
3. If the received packet's byte count is not 0, the byte count amount of data is popped from the receive data FIFO and stored in memory. If the received packet byte count is 0, no data is popped from the receive data FIFO.
4. The receive status readout of the packet of FIFO indicates one of the following:
   a) Global OUT NAK pattern:
   PKTSTS = Global OUT NAK, BCNT = 0x000, EPNUM = (0x0),
   DPID = (0b00).
   These data indicate that the global OUT NAK bit has taken effect.
   b) SETUP packet pattern:
   PKTSTS = SETUP, BCNT = 0x008, EPNUM = Control EP Num,

DPID = DATA0. These data indicate that a SETUP packet for the specified endpoint is now available for reading from the receive FIFO.

c) Setup stage done pattern:
PKTSTS = Setup Stage Done, BCNT = 0x0, EPNUM = Control EP Num, DPID = (0b00).
These data indicate that the setup stage for the specified endpoint has completed and the data stage has started. After this entry is popped from the receive FIFO, the core asserts a setup interrupt on the specified control OUT endpoint.

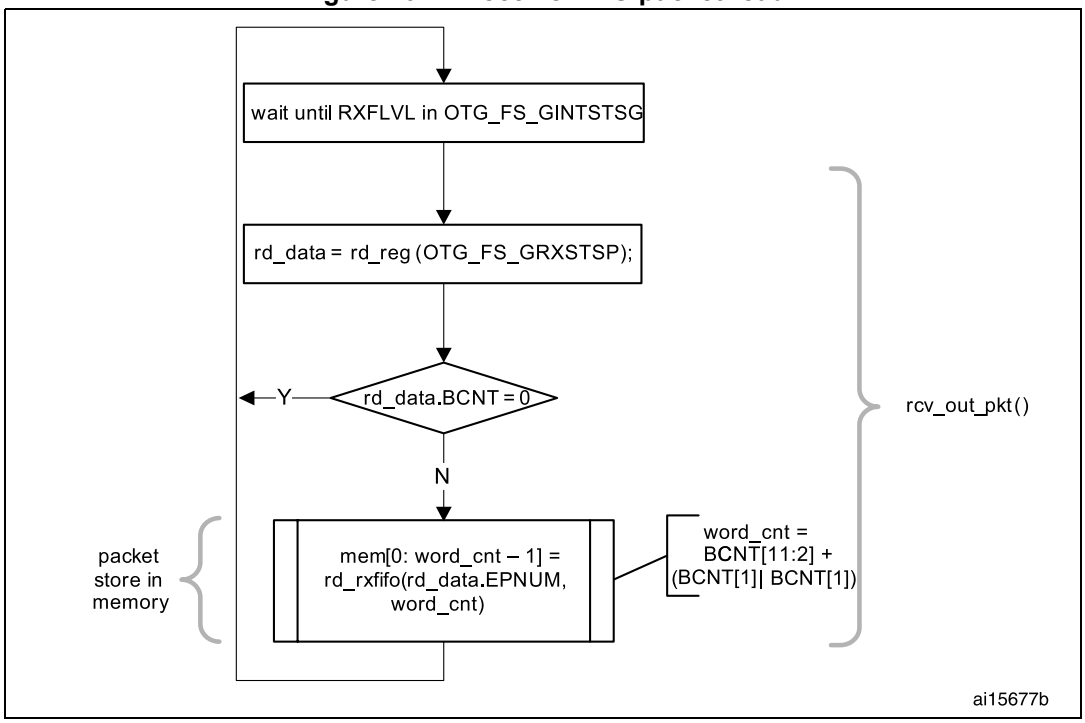
d) Data OUT packet pattern:
PKTSTS = DataOUT, BCNT = size of the received data OUT packet (0 ≤ BCNT ≤ 1 024), EPNUM = EPNUM on which the packet was received, DPID = Actual Data PID.

e) Data transfer completed pattern:
PKTSTS = Data OUT transfer done, BCNT = 0x0, EPNUM = OUT EP Num on which the data transfer is complete, DPID = (0b00).
These data indicate that an OUT data transfer for the specified OUT endpoint has completed. After this entry is popped from the receive FIFO, the core asserts a transfer completed interrupt on the specified OUT endpoint.

5. After the data payload is popped from the receive FIFO, the RXFLVL interrupt (OTG_GINTSTS) must be unmasked.
6. Steps 1–5 are repeated every time the application detects assertion of the interrupt line due to RXFLVL in OTG_GINTSTS. Reading an empty receive FIFO can result in undefined core behavior.

Figure 1027 provides a flowchart of the above procedure.

Figure 1027. Receive FIFO packet read



SETUP transactions

This section describes how the core handles SETUP packets and the application's sequence for handling SETUP transactions.

- **Application requirements**

1. To receive a SETUP packet, the STUPCNT field (OTG_DOEPTSIZx) in a control OUT endpoint must be programmed to a non-zero value. When the application programs the STUPCNT field to a non-zero value, the core receives SETUP packets and writes them to the receive FIFO, irrespective of the NAK status and EPENA bit setting in OTG_DOEPCTLx. The STUPCNT field is decremented every time the control endpoint receives a SETUP packet. If the STUPCNT field is not programmed to a proper value before receiving a SETUP packet, the core still receives the SETUP packet and decrements the STUPCNT field, but the application may not be able to determine the correct number of SETUP packets received in the setup stage of a control transfer.
   – STUPCNT = 3 in OTG_DOEPTSIZx
2. The application must always allocate some extra space in the receive data FIFO, to be able to receive up to three SETUP packets on a control endpoint.
   – The space to be reserved is 13 words. Four words are required for the first SETUP packet, 1 word is required for the setup stage done word and 8 words are required to store two extra SETUP packets among all control endpoints.
   – 4 words per SETUP packet are required to store 8 bytes of SETUP data and 4 bytes of SETUP status (setup packet pattern). The core reserves this space in the receive data FIFO to write SETUP data only, and never uses this space for data packets.
3. The application must read the 2 words of the SETUP packet from the receive FIFO.
4. The application must read and discard the setup stage done word from the receive FIFO.

- **Internal data flow**

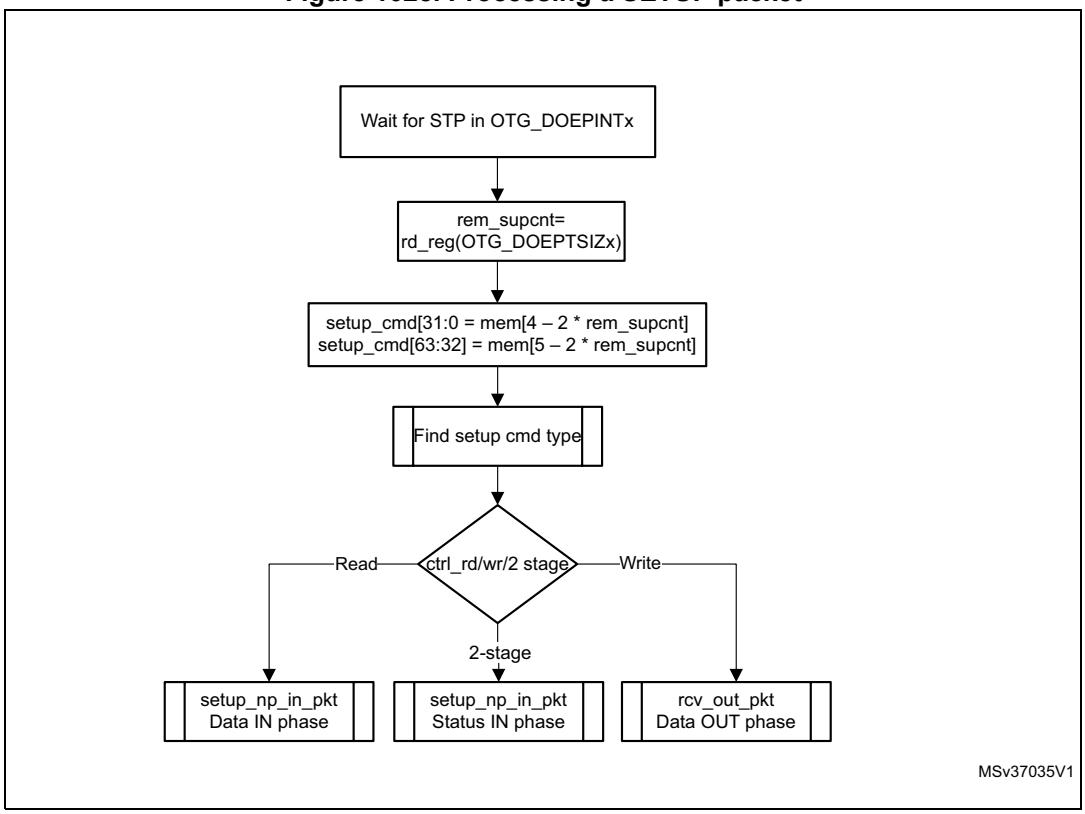
1. When a SETUP packet is received, the core writes the received data to the receive FIFO, without checking for available space in the receive FIFO and irrespective of the endpoint's NAK and STALL bit settings.
   – The core internally sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.
2. For every SETUP packet received on the USB, 3 words of data are written to the receive FIFO, and the STUPCNT field is decremented by 1.
   – The first word contains control information used internally by the core
   – The second word contains the first 4 bytes of the SETUP command
   – The third word contains the last 4 bytes of the SETUP command
3. When the setup stage changes to a data IN/OUT stage, the core writes an entry (setup stage done word) to the receive FIFO, indicating the completion of the setup stage.
4. On the AHB side, SETUP packets are emptied by the application.
5. When the application pops the setup stage done word from the receive FIFO, the core interrupts the application with an STUP interrupt (OTG_DOEPINTx), indicating it can process the received SETUP packet.
6. The core clears the endpoint enable bit for control OUT endpoints.

- **Application programming sequence**

1. Program the OTG_DOEPTSIZx register.
   – STUPCNT = 3
2. Wait for the RXFLVL interrupt (OTG_GINTSTS) and empty the data packets from the receive FIFO.
3. Assertion of the STUP interrupt (OTG_DOEPINTx) marks a successful completion of the SETUP data transfer.
   – On this interrupt, the application must read the OTG_DOEPTSIZx register to determine the number of SETUP packets received and process the last received SETUP packet.

Note: If the application does not enable EP0 before the host sends the SETUP packet, the controller acknowledges (ACKs) the SETUP packet and stores it in the FIFO. However, it does not write to the memory until EP0 is enabled. When the application enables EP0 for the first time and clears the NAK bit while the host sends a DATA OUT packet, the DATA OUT packet is stored in the RxFIFO. The OTG controller then writes the setup data to the memory and disables the endpoint. Although the application expects a Transfer Complete interrupt for the DATA OUT phase, this does not occur because the SETUP packet, rather than the DATA OUT packet, enables EP0 for the first time. As a result, the DATA OUT packet remains in the RxFIFO until the application re-enables EP0. The application must enable EP0 again for the controller to process the DATA OUT packet.

Figure 1028. Processing a SETUP packet



- **Setting the global OUT NAK**

Internal data flow:

1. When the application sets the Global OUT NAK (SGONAK bit in OTG_DCTL), the core stops writing data, except SETUP packets, to the receive FIFO. Irrespective of the space availability in the receive FIFO, non-isochronous OUT tokens receive a NAK handshake response, and the core ignores isochronous OUT data packets
2. The core writes the Global OUT NAK pattern to the receive FIFO. The application must reserve enough receive FIFO space to write this data pattern.
3. When the application pops the Global OUT NAK pattern word from the receive FIFO, the core sets the GONAKEFF interrupt (OTG_GINTSTS).
4. Once the application detects this interrupt, it can assume that the core is in Global OUT NAK mode. The application can clear this interrupt by clearing the SGONAK bit in OTG_DCTL.

Application programming sequence:

1. To stop receiving any kind of data in the receive FIFO, the application must set the Global OUT NAK bit by programming the following field:
   – SGONAK = 1 in OTG_DCTL
2. Wait for the assertion of the GONAKEFF interrupt in OTG_GINTSTS. When asserted, this interrupt indicates that the core has stopped receiving any type of data except SETUP packets.
3. The application can receive valid OUT packets after it has set SGONAK in OTG_DCTL and before the core asserts the GONAKEFF interrupt (OTG_GINTSTS).
4. The application can temporarily mask this interrupt by writing to the GONAKEFFM bit in the OTG_GINTMSK register.
   – GONAKEFFM = 0 in the OTG_GINTMSK register
5. Whenever the application is ready to exit the Global OUT NAK mode, it must clear the SGONAK bit in OTG_DCTL. This also clears the GONAKEFF interrupt (OTG_GINTSTS).
   – CGONAK = 1 in OTG_DCTL
6. If the application has masked this interrupt earlier, it must be unmasked as follows:
   – GONAKEFFM = 1 in OTG_GINTMSK

- **Disabling an OUT endpoint**

The application must use this sequence to disable an OUT endpoint that it has enabled.

Application programming sequence:

1. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core.
   – SGONAK = 1 in OTG_DCTL
2. Wait for the GONAKEFF interrupt (OTG_GINTSTS)
3. Disable the required OUT endpoint by programming the following fields:
   – EPDIS = 1 in OTG_DOEPCTLx
   – SNAK = 1 in OTG_DOEPCTLx
4. Wait for the EPDISD interrupt (OTG_DOEPINTx), which indicates that the OUT endpoint is completely disabled. When the EPDISD interrupt is asserted, the core also clears the following bits:
   – EPDIS = 0 in OTG_DOEPCTLx
   – EPENA = 0 in OTG_DOEPCTLx
5. The application must clear the Global OUT NAK bit to start receiving data from other non-disabled OUT endpoints.
   – SGONAK = 0 in OTG_DCTL

- **Transfer Stop Programming for OUT endpoints**

The application must use the following programing sequence to stop any transfers (because of an interrupt from the host, typically a reset).

Sequence of operations:

1. Enable all OUT endpoints by setting
   – EPENA = 1 in all OTG_DOEPCTLx registers.
2. Flush the RxFIFO as follows
   – Poll OTG_GRSTCTL.AHBIDL until it is 1. This indicates that AHB master is idle.
   – Perform read modify write operation on OTG_GRSTCTL.RXFFLSH =1
   – Poll OTG_GRSTCTL.RXFFLSH until it is 0, but also using a timeout of less than 10 milli-seconds (corresponds to minimum reset signaling duration). If 0 is seen before the timeout, then the RxFIFO flush is successful. If at the moment the timeout occurs, there is still a 1, (this may be due to a packet on EP0 coming from the host) then go back (once only) to the previous step (“Perform read modify write operation”).
3. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core, according to the instructions in “*Setting the global OUT NAK on page 3155*”. This ensures that data in the RxFIFO is sent to the application successfully. Set SGONAK = 1 in OTG_DCTL
4. Wait for the GONAKEFF interrupt (OTG_GINTSTS)
5. Disable all active OUT endpoints by programming the following register bits:
   – EPDIS = 1 in registers OTG_DOEPCTLx
   – SNAK = 1 in registers OTG_DOEPCTLx
6. Wait for the EPDIS interrupt in OTG_DOEPINTx for each OUT endpoint programmed in the previous step. The EPDIS interrupt in OTG_DOEPINTx indicates that the

corresponding OUT endpoint is completely disabled. When the EPDIS interrupt is asserted, the following bits are cleared:

– EPENA = 0 in registers OTG_DOEPCTLx
– EPDIS = 0 in registers OTG_DOEPCTLx
– SNAK = 0 in registers OTG_DOEPCTLx

- **Generic non-isochronous OUT data transfers**

This section describes a regular non-isochronous OUT data transfer (control, bulk, or interrupt).

Application requirements:

1. Before setting up an OUT transfer, the application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer.
2. For OUT transfers, the transfer size field in the endpoint's transfer size register must be a multiple of the maximum packet size of the endpoint, adjusted to the word boundary.
   – transfer size[EPNUM] = *n* × (MPSIZ[EPNUM] + 4 – (MPSIZ[EPNUM] mod 4))
   – packet count[EPNUM] = *n*
   – *n* > 0
3. On any OUT endpoint interrupt, the application must read the endpoint's transfer size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
   – Payload size in memory = application programmed initial transfer size – core updated final transfer size
   – Number of USB packets in which this payload was received = application programmed initial packet count – core updated final packet count

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the packet count field for that endpoint by 1.
   – OUT data packets received with bad data CRC are flushed from the receive FIFO automatically.
   – After sending an ACK for the packet on the USB, the core discards non-isochronous OUT data packets that the host, which cannot detect the ACK, re-sends. The application does not detect multiple back-to-back data OUT packets on the same endpoint with the same data PID. In this case the packet count is not decremented.
   – If there is no space in the receive FIFO, isochronous or non-isochronous data packets are ignored and not written to the receive FIFO. Additionally, non-isochronous OUT tokens receive a NAK handshake reply.
   – In all the above three cases, the packet count is not decremented because no data are written to the receive FIFO.

3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or non-isochronous data packets are ignored and not written to the receive FIFO, and non-isochronous OUT tokens receive a NAK handshake reply.
4. After the data are written to the receive FIFO, the application reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
5. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.
6. The OUT data transfer completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions:
   – The transfer size is 0 and the packet count is 0
   – The last OUT data packet written to the receive FIFO is a short packet (0 ≤ packet size < maximum packet size)
7. When either the application pops this entry (OUT data transfer completed), a transfer completed interrupt is generated for the endpoint and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_DOEPTSIZx register for the transfer size and the corresponding packet count.
2. Program the OTG_DOEPCTLx register with the endpoint characteristics, and set the EPENA and CNAK bits.
   – EPENA = 1 in OTG_DOEPCTLx
   – CNAK = 1 in OTG_DOEPCTLx
3. Wait for the RXFLVL interrupt (in OTG_GINTSTS) and empty the data packets from the receive FIFO.
   – This step can be repeated many times, depending on the transfer size.
4. Asserting the XFRC interrupt (OTG_DOEPINTx) marks a successful completion of the non-isochronous OUT data transfer.
5. Read the OTG_DOEPTSIZx register to determine the size of the received data payload.

- **Generic isochronous OUT data transfer**

This section describes a regular isochronous OUT data transfer.

Application requirements:

1. All the application requirements for non-isochronous OUT data transfers also apply to isochronous OUT data transfers.
2. For isochronous OUT data transfers, the transfer size and packet count fields must always be set to the number of maximum-packet-size packets that can be received in a single frame and no more. Isochronous OUT data transfers cannot span more than 1 frame.
3. The application must read all isochronous OUT data packets from the receive FIFO (data and status) before the end of the periodic frame (EOPF interrupt in OTG_GINTSTS).
4. To receive data in the following frame, an isochronous OUT endpoint must be enabled after the EOPF (OTG_GINTSTS) and before the SOF (OTG_GINTSTS).

Internal data flow:

1. The internal data flow for isochronous OUT endpoints is the same as that for non-isochronous OUT endpoints, but for a few differences.
2. When an isochronous OUT endpoint is enabled by setting the endpoint enable and clearing the NAK bits, the Even/Odd frame bit must also be set appropriately. The core receives data on an isochronous OUT endpoint in a particular frame only if the following condition is met:
   – EONUM (in OTG_DOEPCTLx) = FNSOF[0] (in OTG_DSTS)
3. When the application completely reads an isochronous OUT data packet (data and status) from the receive FIFO, the core updates the RXDPID field in OTG_DOEPTSIZx with the data PID of the last isochronous OUT data packet read from the receive FIFO.

Application programming sequence:

1. Program the OTG_DOEPTSIZx register for the transfer size and the corresponding packet count
2. Program the OTG_DOEPCTLx register with the endpoint characteristics and set the endpoint enable, ClearNAK, and Even/Odd frame bits.
   – EPENA = 1
   – CNAK = 1
   – EONUM = (0: Even/1: Odd)
3. Wait for the RXFLVL interrupt (in OTG_GINTSTS) and empty the data packets from the receive FIFO
   – This step can be repeated many times, depending on the transfer size.
4. The assertion of the XFRC interrupt (in OTG_DOEPINTx) marks the completion of the isochronous OUT data transfer. This interrupt does not necessarily mean that the data in memory are good.
5. This interrupt cannot always be detected for isochronous OUT transfers. Instead, the application can detect the INCOMPISOOUT interrupt in OTG_GINTSTS.
6. Read the OTG_DOEPTSIZx register to determine the size of the received transfer and to determine the validity of the data received in the frame. The application must treat the data received in memory as valid only if one of the following conditions is met:
   – RXDPID = DATA0 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 1
   – RXDPID = DATA1 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 2

   The number of USB packets in which this payload was received = Application programmed initial packet count – core updated final packet count

   The application can discard invalid data packets.

- **Incomplete isochronous OUT data transfers**

This section describes the application programming sequence when isochronous OUT data packets are dropped inside the core.

Internal data flow:

1. For isochronous OUT endpoints, the XFRC interrupt (in OTG_DOEPINTx) may not always be asserted. If the core drops isochronous OUT data packets, the application

could fail to detect the XFRC interrupt (OTG_DOEPINTx) under the following circumstances:

- When the receive FIFO cannot accommodate the complete ISO OUT data packet, the core drops the received ISO OUT data
- When the isochronous OUT data packet is received with CRC errors
- When the isochronous OUT token received by the core is corrupted
- When the application is very slow in reading the data from the receive FIFO

2. When the core detects an end of periodic frame before transfer completion to all isochronous OUT endpoints, it asserts the incomplete isochronous OUT data interrupt (INCOMPISOOUT in OTG_GINTSTS), indicating that an XFRC interrupt (in OTG_DOEPINTx) is not asserted on at least one of the isochronous OUT endpoints. At this point, the endpoint with the incomplete transfer remains enabled, but no active transfers remain in progress on this endpoint on the USB.

Application programming sequence:

1. Asserting the INCOMPISOOUT interrupt (OTG_GINTSTS) indicates that in the current frame, at least one isochronous OUT endpoint has an incomplete transfer.
2. If this occurs because isochronous OUT data is not completely emptied from the endpoint, the application must ensure that the application empties all isochronous OUT data (data and status) from the receive FIFO before proceeding.
   - When all data are emptied from the receive FIFO, the application can detect the XFRC interrupt (OTG_DOEPINTx). In this case, the application must re-enable the endpoint to receive isochronous OUT data in the next frame.
3. When it receives an INCOMPISOOUT interrupt (in OTG_GINTSTS), the application must read the control registers of all isochronous OUT endpoints (OTG_DOEPCTLx) to determine which endpoints had an incomplete transfer in the current microframe. An endpoint transfer is incomplete if both the following conditions are met:
   - EONUM bit (in OTG_DOEPCTLx) = FNSOF[0] (in OTG_DSTS)
   - EPENA = 1 (in OTG_DOEPCTLx)
4. The previous step must be performed before the SOF interrupt (in OTG_GINTSTS) is detected, to ensure that the current frame number is not changed.
5. For isochronous OUT endpoints with incomplete transfers, the application must discard the data in the memory and disable the endpoint by setting the EPDIS bit in OTG_DOEPCTLx.
6. Wait for the EPDISD interrupt (in OTG_DOEPINTx) and enable the endpoint to receive new data in the next frame.
   - Because the core can take some time to disable the endpoint, the application may not be able to receive the data in the next frame after receiving bad isochronous data.

- **Stalling a non-isochronous OUT endpoint**

This section describes how the application can stall a non-isochronous endpoint.

1. Put the core in the Global OUT NAK mode.
2. Disable the required endpoint
   – When disabling the endpoint, instead of setting the SNAK bit in OTG_DOEPCTL, set STALL = 1 (in OTG_DOEPCTL).

     The STALL bit always takes precedence over the NAK bit.
3. When the application is ready to end the STALL handshake for the endpoint, the STALL bit (in OTG_DOEPCTLx) must be cleared.
4. If the application is setting or clearing a STALL for an endpoint due to a SetFeature.Endpoint Halt or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the status stage transfer on the control endpoint.

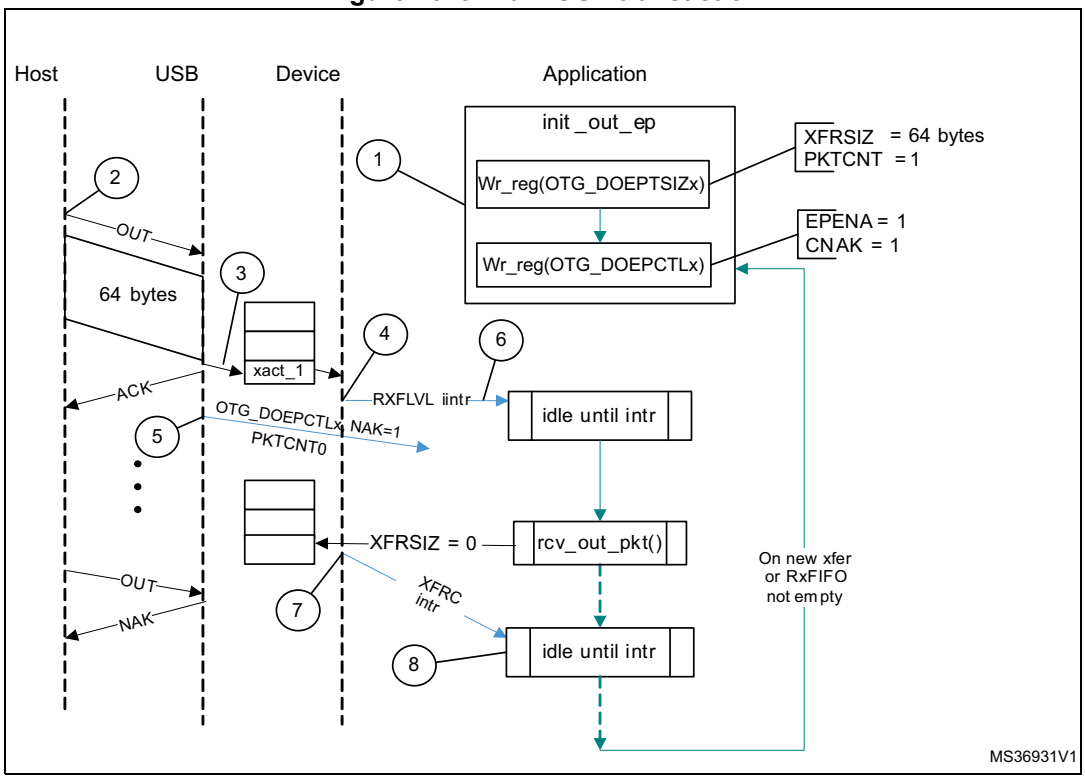
Examples

This section describes and depicts some fundamental transfer types and scenarios.

- Bulk OUT transaction

Figure 1029 depicts the reception of a single Bulk OUT data packet from the USB to the AHB and describes the events involved in the process.

Figure 1029. Bulk OUT transaction



After a SetConfiguration/SetInterface command, the application initializes all OUT endpoints by setting CNAK = 1 and EPENA = 1 (in OTG_DOEPCTLx), and setting a suitable XFRSIZ and PKTCNT in the OTG_DOEPTSIZx register.

1. host attempts to send data (OUT token) to an endpoint.
2. When the core receives the OUT token on the USB, it stores the packet in the Rx FIFO because space is available there.
3. After writing the complete packet in the Rx FIFO, the core then asserts the RXFLVL interrupt (in OTG_GINTSTS).
4. On receiving the PKTCNT number of USB packets, the core internally sets the NAK bit for this endpoint to prevent it from receiving any more packets.
5. The application processes the interrupt and reads the data from the Rx FIFO.
6. When the application has read all the data (equivalent to XFRSIZ), the core generates an XFRC interrupt (in OTG_DOEPINTx).
7. The application processes the interrupt and uses the setting of the XFRC interrupt bit (in OTG_DOEPINTx) to determine that the intended transfer is complete.

IN data transfers

- **Packet write**

This section describes how the application writes data packets to the endpoint FIFO when dedicated transmit FIFOs are enabled.

1. The application can either choose the polling or the interrupt mode.
   - In polling mode, the application monitors the status of the endpoint transmit data FIFO by reading the OTG_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   - In interrupt mode, the application waits for the TXFE interrupt (in OTG_DIEPINTx) and then reads the OTG_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   - To write a single non-zero length data packet, there must be space to write the entire packet in the data FIFO.
   - To write zero length packet, the application must not look at the FIFO space.
2. Using one of the above mentioned methods, when the application determines that there is enough space to write a transmit packet, the application must first write into the endpoint control register, before writing the data into the data FIFO. Typically, the application, must do a read modify write on the OTG_DIEPCTLx register to avoid modifying the contents of the register, except for setting the endpoint enable bit.

The application can write multiple packets for the same endpoint into the transmit FIFO, if space is available. For periodic IN endpoints, the application must write packets only for one microframe. It can write packets for the next periodic transaction only after getting transfer complete for the previous transaction.

- **Setting IN endpoint NAK**

Internal data flow:

1. When the application sets the IN NAK for a particular endpoint, the core stops transmitting data on the endpoint, irrespective of data availability in the endpoint's transmit FIFO.
2. Non-isochronous IN tokens receive a NAK handshake reply
   – Isochronous IN tokens receive a zero-data-length packet reply
3. The core asserts the INEPNE (IN endpoint NAK effective) interrupt in OTG_DIEPINTx in response to the SNAK bit in OTG_DIEPCTLx.
4. Once this interrupt is seen by the application, the application can assume that the endpoint is in IN NAK mode. This interrupt can be cleared by the application by setting the CNAK bit in OTG_DIEPCTLx.

Application programming sequence:

1. To stop transmitting any data on a particular IN endpoint, the application must set the IN NAK bit. To set this bit, the following field must be programmed.
   – SNAK = 1 in OTG_DIEPCTLx
2. Wait for assertion of the INEPNE interrupt in OTG_DIEPINTx. This interrupt indicates that the core has stopped transmitting data on the endpoint.
3. The core can transmit valid IN data on the endpoint after the application has set the NAK bit, but before the assertion of the NAK Effective interrupt.
4. The application can mask this interrupt temporarily by writing to the INEPNEM bit in OTG_DIEPMSK.
   – INEPNEM = 0 in OTG_DIEPMSK
5. To exit endpoint NAK mode, the application must clear the NAK status bit (NAKSTS) in OTG_DIEPCTLx. This also clears the INEPNE interrupt (in OTG_DIEPINTx).
   – CNAK = 1 in OTG_DIEPCTLx
6. If the application masked this interrupt earlier, it must be unmasked as follows:
   – INEPNEM = 1 in OTG_DIEPMSK

- **IN endpoint disable**

Use the following sequence to disable a specific IN endpoint that has been previously enabled.

Application programming sequence:

1. The application must stop writing data on the AHB for the IN endpoint to be disabled.
2. The application must set the endpoint in NAK mode.
   – SNAK = 1 in OTG_DIEPCTLx
3. Wait for the INEPNE interrupt in OTG_DIEPINTx.
4. Set the following bits in the OTG_DIEPCTLx register for the endpoint that must be disabled.
   – EPDIS = 1 in OTG_DIEPCTLx
   – SNAK = 1 in OTG_DIEPCTLx
5. Assertion of the EPDISD interrupt in OTG_DIEPINTx indicates that the core has completely disabled the specified endpoint. Along with the assertion of the interrupt, the core also clears the following bits:
   – EPENA = 0 in OTG_DIEPCTLx
   – EPDIS = 0 in OTG_DIEPCTLx
6. The application must read the OTG_DIEPTSIZx register for the periodic IN EP, to calculate how much data on the endpoint were transmitted on the USB.
7. The application must flush the data in the endpoint transmit FIFO, by setting the following fields in the OTG_GRSTCTL register:
   – TXFNUM (in OTG_GRSTCTL) = Endpoint transmit FIFO number
   – TXFFLSH in (OTG_GRSTCTL) = 1

The application must poll the OTG_GRSTCTL register, until the TXFFLSH bit is cleared by the core, which indicates the end of flush operation. To transmit new data on this endpoint, the application can re-enable the endpoint at a later point.

- **Transfer Stop Programming for IN endpoints**

The application must use the following programing sequence to stop any transfers (because of an interrupt from the host, typically a reset).

Sequence of operations:

1. Disable the IN endpoint by setting:
   – EPDIS = 1 in all OTG_DIEPCTLx registers
2. Wait for the EPDIS interrupt in OTG_DIEPINTx, which indicates that the IN endpoint is completely disabled. When the EPDIS interrupt is asserted the following bits are cleared:
   – EPDIS = 0 in OTG_DIEPCTLx
   – EPENA = 0 in OTG_DIEPCTLx
3. Flush the TxFIFO by programming the following bits:
   – TXFFLSH = 1 in OTG_GRSTCTL
   – TXFNUM = “FIFO number specific to endpoint” in OTG_GRSTCTL
4. The application can start polling till TXFFLSH in OTG_GRSTCTL is cleared. When this bit is cleared, it ensures that there is no data left in the Tx FIFO.

- **Generic non-periodic IN data transfers**

Application requirements:

1. Before setting up an IN transfer, the application must ensure that all data to be transmitted as part of the IN transfer are part of a single buffer.
2. For IN transfers, the transfer size field in the endpoint transfer size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
   – To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:

     Transfer size[EPNUM] = *x* × MPSIZ[EPNUM] + sp

     If (sp > 0), then packet count[EPNUM] = x + 1.
     Otherwise, packet count[EPNUM] = *x*
   – To transmit a single zero-length data packet:

     Transfer size[EPNUM] = 0

     Packet count[EPNUM] = 1
   – To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer into two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.

     First transfer: transfer size[EPNUM] = *x* × MPSIZ[epnum]; packet count = *n*;

     Second transfer: transfer size[EPNUM] = 0; packet count = 1;
3. Once an endpoint is enabled for data transfers, the core updates the transfer size register. At the end of the IN transfer, the application must read the transfer size register to determine how much data posted in the transmit FIFO have already been sent on the USB.
4. Data fetched into transmit FIFO = Application-programmed initial transfer size – core-updated final transfer size
   – Data transmitted on USB = (application-programmed initial packet count – core updated final packet count) × MPSIZ[EPNUM]
   – Data yet to be transmitted on USB = (Application-programmed initial transfer size – data transmitted on USB)

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the transmit FIFO for the endpoint.
3. Every time a packet is written into the transmit FIFO by the application, the transfer size for that endpoint is decremented by the packet size. The data is fetched from the memory by the application, until the transfer size for the endpoint becomes 0. After writing the data into the FIFO, the “number of packets in FIFO” count is incremented (this is a 3-bit count, internally maintained by the core for each IN endpoint transmit FIFO. The maximum number of packets maintained by the core at any time in an IN endpoint FIFO is eight). For zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.
4. Once the data are written to the transmit FIFO, the core reads them out upon receiving an IN token. For every non-isochronous IN data packet transmitted with an ACK

handshake, the packet count for the endpoint is decremented by one, until the packet count is zero. The packet count is not decremented on a timeout.

5. For zero length packets (indicated by an internal zero length flag), the core sends out a zero-length packet for the IN token and decrements the packet count field.
6. If there are no data in the FIFO for a received IN token and the packet count field for that endpoint is zero, the core generates an “IN token received when Tx FIFO is empty” (ITTXFE) interrupt for the endpoint, provided that the endpoint NAK bit is not set. The core responds with a NAK handshake for non-isochronous endpoints on the USB.
7. The core internally rewinds the FIFO pointers and no timeout interrupt is generated.
8. When the transfer size is 0 and the packet count is 0, the transfer complete (XFRC) interrupt for the endpoint is generated and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_DIEPTSIZx register with the transfer size and corresponding packet count.
2. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA (endpoint enable) bits.
3. When transmitting non-zero length data packet, the application must poll the OTG_DTXFSTSx register (where x is the FIFO number associated with that endpoint) to determine whether there is enough space in the data FIFO. The application can optionally use TXFE (in OTG_DIEPINTx) before writing the data.

- **Generic periodic IN data transfers**

This section describes a typical periodic IN data transfer.

Application requirements:

1. Application requirements 1, 2, 3, and 4 of *Generic non-periodic IN data transfers on page 3165* also apply to periodic IN data transfers, except for a slight modification of requirement 2.
   – The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To

transmit a few maximum-packet-size packets and a short packet at the end of the transfer, the following conditions must be met:

transfer size[EPNUM] = x × MPSIZ[EPNUM] + sp
(where *x* is an integer ≥ 0, and 0 ≤ sp < MPSIZ[EPNUM])

If (sp > 0), packet count[EPNUM] = *x* + 1
Otherwise, packet count[EPNUM] = *x*;

MCNT[EPNUM] = packet count[EPNUM]

– The application cannot transmit a zero-length data packet at the end of a transfer. It can transmit a single zero-length data packet by itself. To transmit a single zero-length data packet:
– transfer size[EPNUM] = 0

packet count[EPNUM] = 1

MCNT[EPNUM] = packet count[EPNUM]

2. The application can only schedule data transfers one frame at a time.
   – (MCNT – 1) × MPSIZ ≤ XFERSIZ ≤ MCNT × MPSIZ
   – PKTCNT = MCNT (in OTG_DIEPTSIZx)
   – If XFERSIZ < MCNT × MPSIZ, the last data packet of the transfer is a short packet.
   – Note that: MCNT is in OTG_DIEPTSIZx, MPSIZ is in OTG_DIEPCTLx, PKTCNT is in OTG_DIEPTSIZx and XFERSIZ is in OTG_DIEPTSIZx
3. The complete data to be transmitted in the frame must be written into the transmit FIFO by the application, before the IN token is received. Even when 1 word of the data to be transmitted per frame is missing in the transmit FIFO when the IN token is received, the core behaves as when the FIFO is empty. When the transmit FIFO is empty:
   – A zero data length packet would be transmitted on the USB for isochronous IN endpoints
   – A NAK handshake would be transmitted on the USB for interrupt IN endpoints

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the associated transmit FIFO for the endpoint.
3. Every time the application writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data are fetched from application memory until the transfer size for the endpoint becomes 0.
4. When an IN token is received for a periodic endpoint, the core transmits the data in the FIFO, if available. If the complete data payload (complete packet, in dedicated FIFO

mode) for the frame is not present in the FIFO, then the core generates an IN token received when Tx FIFO empty interrupt for the endpoint.
   – A zero-length data packet is transmitted on the USB for isochronous IN endpoints
   – A NAK handshake is transmitted on the USB for interrupt IN endpoints
5. The packet count for the endpoint is decremented by 1 under the following conditions:
   – For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
   – For interrupt endpoints, when an ACK handshake is transmitted
   – When the transfer size and packet count are both 0, the transfer completed interrupt for the endpoint is generated and the endpoint enable is cleared.
6. At the “Periodic frame Interval” (controlled by PFIVL in OTG_DCFG), when the core finds non-empty any of the isochronous IN endpoint FIFOs scheduled for the current frame non-empty, the core generates an IISOIXFR interrupt in OTG_GINTSTS.

Application programming sequence:

1. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA bits.
2. Write the data to be transmitted in the next frame to the transmit FIFO.
3. Asserting the ITTXFE interrupt (in OTG_DIEPINTx) indicates that the application has not yet written all data to be transmitted to the transmit FIFO.
4. If the interrupt endpoint is already enabled when this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint so that the data can be transmitted on the next IN token attempt.
5. Asserting the XFRC interrupt (in OTG_DIEPINTx) with no ITTXFE interrupt in OTG_DIEPINTx indicates the successful completion of an isochronous IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
6. Asserting the XFRC interrupt (in OTG_DIEPINTx), with or without the ITTXFE interrupt (in OTG_DIEPINTx), indicates the successful completion of an interrupt IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
7. Asserting the incomplete isochronous IN transfer (IISOIXFR) interrupt in OTG_GINTSTS with none of the aforementioned interrupts indicates the core did not receive at least 1 periodic IN token in the current frame.

- **Incomplete isochronous IN data transfers**

This section describes what the application must do on an incomplete isochronous IN data transfer.

Internal data flow:

1. An isochronous IN transfer is treated as incomplete in one of the following conditions:
   a) The core receives a corrupted isochronous IN token on at least one isochronous IN endpoint. In this case, the application detects an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS).
   b) The application is slow to write the complete data payload to the transmit FIFO and an IN token is received before the complete data payload is written to the FIFO. In this case, the application detects an IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx. The application can ignore this interrupt, as it

eventually results in an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS) at the end of periodic frame.

The core transmits a zero-length data packet on the USB in response to the received IN token.

2. The application must stop writing the data payload to the transmit FIFO as soon as possible.
3. The application must set the NAK bit and the disable bit for the endpoint.
4. The core disables the endpoint, clears the disable bit, and asserts the endpoint disable interrupt for the endpoint.

Application programming sequence:

1. The application can ignore the IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx on any isochronous IN endpoint, as it eventually results in an incomplete isochronous IN transfer interrupt (in OTG_GINTSTS).
2. Assertion of the incomplete isochronous IN transfer interrupt (in OTG_GINTSTS) indicates an incomplete isochronous IN transfer on at least one of the isochronous IN endpoints.
3. The application must read the endpoint control register for all isochronous IN endpoints to detect endpoints with incomplete IN data transfers.
4. The application must stop writing data to the Periodic Transmit FIFOs associated with these endpoints on the AHB.
5. Program the following fields in the OTG_DIEPCTLx register to disable the endpoint:
   – SNAK = 1 in OTG_DIEPCTLx
   – EPDIS = 1 in OTG_DIEPCTLx
6. The assertion of the endpoint disabled interrupt in OTG_DIEPINTx indicates that the core has disabled the endpoint.
   – At this point, the application must flush the data in the associated transmit FIFO or overwrite the existing data in the FIFO by enabling the endpoint for a new transfer in the next microframe. To flush the data, the application must use the OTG_GRSTCTL register.

- **Stalling non-isochronous IN endpoints**

This section describes how the application can stall a non-isochronous endpoint.

Application programming sequence:

1. Disable the IN endpoint to be stalled. Set the STALL bit as well.
2. EPDIS = 1 in OTG_DIEPCTLx, when the endpoint is already enabled
   – STALL = 1 in OTG_DIEPCTLx
   – The STALL bit always takes precedence over the NAK bit
3. Assertion of the endpoint disabled interrupt (in OTG_DIEPINTx) indicates to the application that the core has disabled the specified endpoint.
4. The application must flush the non-periodic or periodic transmit FIFO, depending on the endpoint type. In case of a non-periodic endpoint, the application must re-enable the other non-periodic endpoints that do not need to be stalled, to transmit data.
5. Whenever the application is ready to end the STALL handshake for the endpoint, the STALL bit must be cleared in OTG_DIEPCTLx.
6. If the application sets or clears a STALL bit for an endpoint due to a SetFeature.Endpoint Halt command or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the status stage transfer on the control endpoint.

Special case: stalling the control OUT endpoint

The core must stall IN/OUT tokens if, during the data stage of a control transfer, the host sends more IN/OUT tokens than are specified in the SETUP packet. In this case, the application must enable the ITTXFE interrupt in OTG_DIEPINTx and the OTEPDIS interrupt in OTG_DOEPINTx during the data stage of the control transfer, after the core has transferred the amount of data specified in the SETUP packet. Then, when the application receives this interrupt, it must set the STALL bit in the corresponding endpoint control register, and clear this interrupt.

50.16.6 Worst case response time

When the OTG_FS controller acts as a device, there is a worst case response time for any tokens that follow an isochronous OUT. This worst case response time depends on the AHB clock frequency.

The core registers are in the AHB domain, and the core does not accept another token before updating these register values. The worst case is for any token following an isochronous OUT, because for an isochronous transaction, there is no handshake and the next token could come sooner. This worst case value is 7 PHY clocks when the AHB clock is the same as the PHY clock. When the AHB clock is faster, this value is smaller.

If this worst case condition occurs, the core responds to bulk/interrupt tokens with a NAK and drops isochronous and SETUP tokens. The host interprets this as a timeout condition for SETUP and retries the SETUP packet. For isochronous transfers, the Incomplete isochronous IN transfer interrupt (IISOIXFR) and Incomplete isochronous OUT transfer interrupt (IISOOXFR) inform the application that isochronous IN/OUT packets were dropped.

Choosing the value of TRDT in OTG_GUSBCFG

The value in TRDT (OTG_GUSBCFG) is the time it takes for the MAC, in terms of PHY clocks after it has received an IN token, to get the FIFO status, and thus the first data from the PFC block. This time involves the synchronization delay between the PHY and AHB clocks. The worst case delay for this is when the AHB clock is the same as the PHY clock. In this case, the delay is 5 clocks.

Once the MAC receives an IN token, this information (token received) is synchronized to the AHB clock by the PFC (the PFC runs on the AHB clock). The PFC then reads the data from the SPRAM and writes them into the dual clock source buffer. The MAC then reads the data out of the source buffer (4 deep).

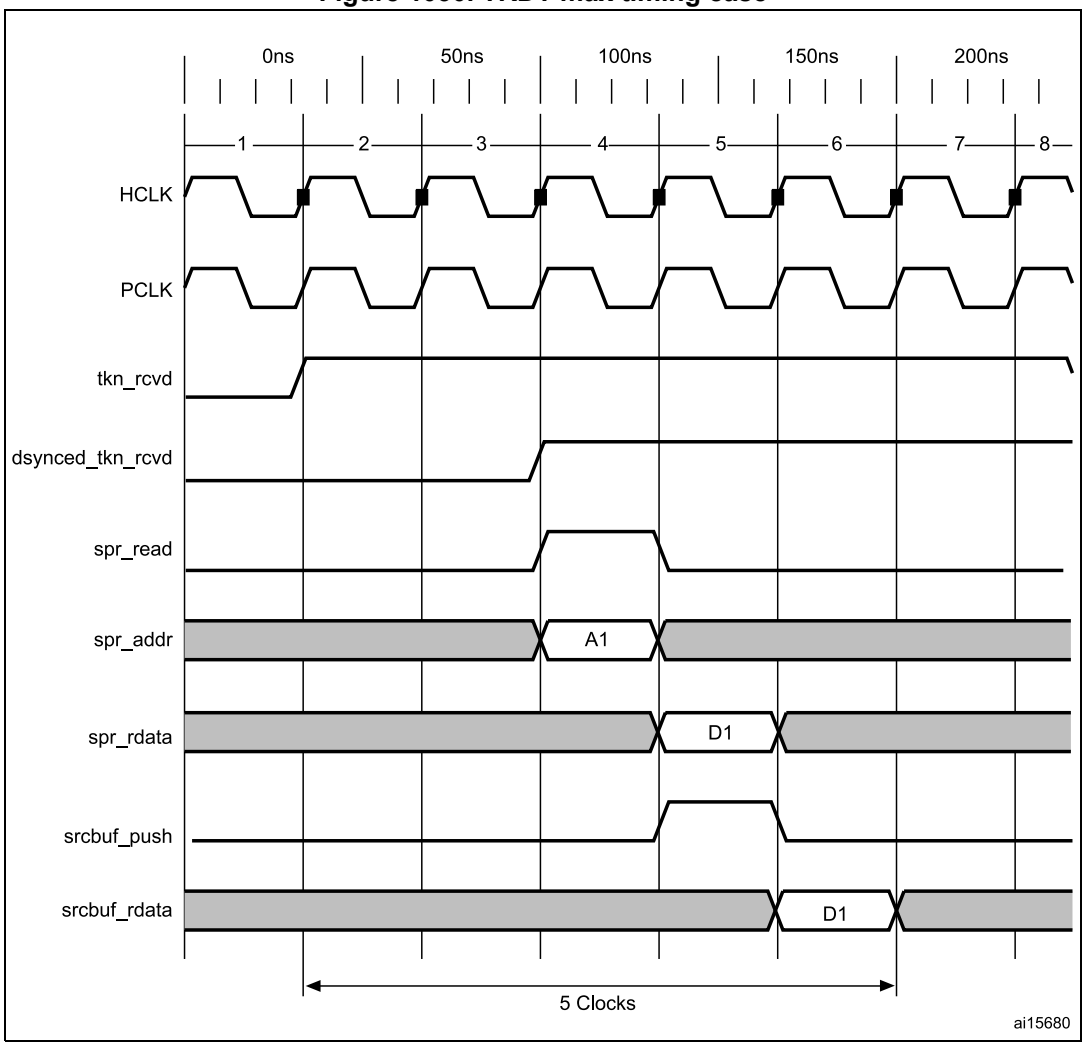
If the AHB is running at a higher frequency than the PHY, the application can use a smaller value for TRDT (in OTG_GUSBCFG).

Figure 1030 has the following signals:

- tkn_rcvd: Token received information from MAC to PFC
- dynced_tkn_rcvd: Doubled sync tkn_rcvd, from PCLK to HCLK domain
- spr_read: Read to SPRAM
- spr_addr: Address to SPRAM
- spr_rdata: Read data from SPRAM
- srcbuf_push: Push to the source buffer
- srcbuf_rdata: Read data from the source buffer. Data seen by MAC

To calculate the value of TRDT, refer to *Table 589: TRDT values*.

Figure 1030. TRDT max timing case



50.16.7 OTG programming model

The OTG_FS controller is an OTG device. When the core is connected to an “A” plug, it is referred to as an A-device. When the core is connected to a “B” plug it is referred to as a B-device.

51 USB Type-C®/USB Power Delivery interface (UCPD)

51.1 UCPD introduction

The USB Type-C/USB Power Delivery interface complies with:

- Universal Serial Bus Type-C Cable and Connector Specification: release 2.4, Oct 2024
- Universal Serial Bus Power Delivery specifications:
  - revision 2.0, version 1.3, January 12, 2017
  - revision 3.2, version 1.1, October 2024

It integrates the physical layer of the Power Delivery (PD) specification, with CC signaling method (no VBUS), for operation with Type-C cables.

51.2 UCPD main features

- Compliance with USB Type-C specification release 2.4
- Compliance with USB Power Delivery specifications revision 2.0 and 3.2
  - Enabling advanced applications such as PPS (programmable power supply)
- Stop mode low-power operation support
- Built-in analog PHY
  - USB Type-C pull-up (Rp, all values) and pull-down (Rd) resistors
  - “Dead battery” Rd support
  - USB Power Delivery message transmission and reception
  - FRS (fast role swap) Rx support
- Digital controller
  - BMC (bi-phase mark coding) encode and decode
  - 4b5b encode and decode
  - USB Type-C level detection with de-bounce, generating interrupts
  - FRS signaling
  - FRS detection, generating an interrupt
  - DMA-compatible byte-level interface for USB Power Delivery payload, generating interrupts
  - USB Power Delivery clock pre-scaler / dividers
  - CRC generation/checking
  - Support of ordered sets, with a programmable ordered set mask at receive
  - Clock recovery from incoming Rx stream

51.3 UCPD implementation

The devices have one UCPD controller to support one USB Type-C port.

In the following table, “X” denotes *supported*, and “-” *not supported*.

Table 592. UCPD implementation

UCPD feature	UCPD1
Dead battery support via UCPDx_DBCC1 and UCPDx_DBCC2 external signals	X
UCPDx_FRSTX1/2 as alternate function pins	X
Fully automatic trimming	X(1)
USB PD receiver hardware filter control	X
Discrete-component PHY support	-

1. No software trimming required.

51.4 UCPD functional description

The UCPD peripheral provides hardware support for the USB Power Delivery control interface specification, using I/Os specifically designed for that purpose.

The built-in PHY directly detects Type-C voltage levels, supports Power Delivery BIST carrier mode 2 (Tx only), BIST test data (Tx and Rx), and Power Delivery Rx FRS signaling.

For Power Delivery FRS Tx signaling, the device can be configured to control, through UCPD_FRSTX1/2 pins (alternate functions), external NMOS transistors that ensure low-resistance pull-down on CC lines.

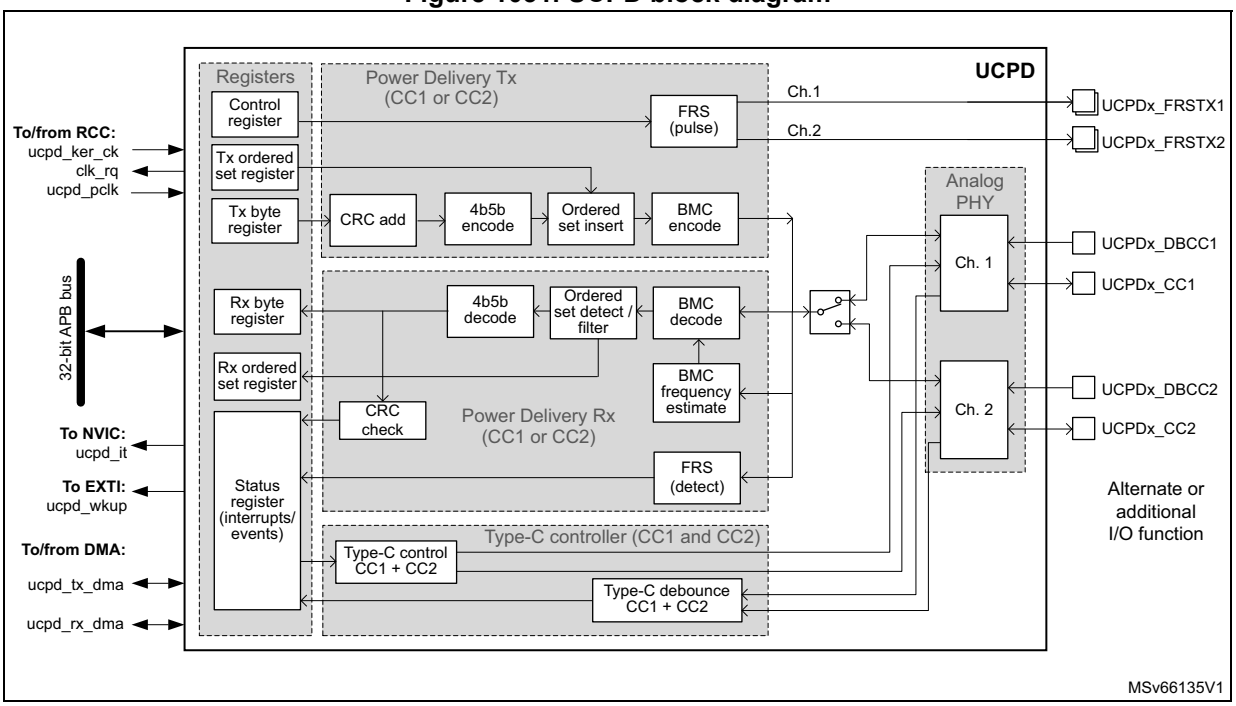
The UCPD transmitter BMC (bi-phase mark) encodes and transmits data: preamble, SOP, payload data from protocol layer (after 4b5b-encoding), CRC, and EOP on the Type-C connector CC lines. It automatically inserts inter-frame gap and executes “Hard Reset”.

The UCPD receiver detects SOP, BMC-decodes the incoming stream, recovers the preamble, 4b5b-decodes payload data, detects EOP, and checks CRC. It automatically detects five K-code SOP and two Reset ordered sets, plus two software-defined patterns (allows for only three out of four K-codes being correctly received, as defined by the standard).

In Stop mode, the peripheral maintains the ability to detect incoming USB Power Delivery messages and FRS signaling, which allows low-power operation.

51.4.1 UCPD block diagram

Figure 1031. UCPD block diagram



The following table lists external signals (alternate or additional I/O functions).

Table 593. UCPD signals on pins

Pin name	Signal type	Description
UCPDx_FRSTX1	Output	USB Type-C fast role swap (FRS) signaling, applicable to DRPs only. The signal (active high) drives an external NMOS transistor that pulls down the CC1 line.
UCPDx_FRSTX2	Output	USB Type-C fast role swap (FRS) signaling, applicable to DRPs only. The signal (active high) drives an external NMOS transistor that pulls down the CC2 line.
UCPDx_CC1	Input/output	USB Type-C configuration control line 1, to be routed to the USB Type-C connector CC1 terminal.
UCPDx_CC2	Input/output	USB Type-C configuration control line 2, to be routed to the USB Type-C connector CC2 terminal.
UCPDx_DBCC1	Input	USB Type-C configuration control line 1 dead battery signal, to be routed to the USB Type-C connector CC1 terminal if dead battery support is required.
UCPDx_DBCC2	Input	USB Type-C configuration control line 2 dead battery signal, to be routed to the USB Type-C connector CC2 terminal if dead battery support is required.

The following table lists key internal signals.

Table 594. UCPD internal signals

Internal signal name	Signal type	Description
ucpd_pclk	Input	APB clock for registers
ucpd_ker_ck	Input	Kernel clock
ucpd_tx_dma	Input/Output	Rx DMA acknowledge / request
ucpd_rx_dma	Input/Output	Tx DMA acknowledge / request
ucpd_it	Output	Interrupt request (all interrupts OR-ed) connected to NVIC
ucpd_wkup	Output	Wake-up request connected to EXTI
clk_rq	Output	Clock request connected to RCC

51.4.2 UCPD reset and clocks

The peripheral has a single reset signal (APB bus reset).

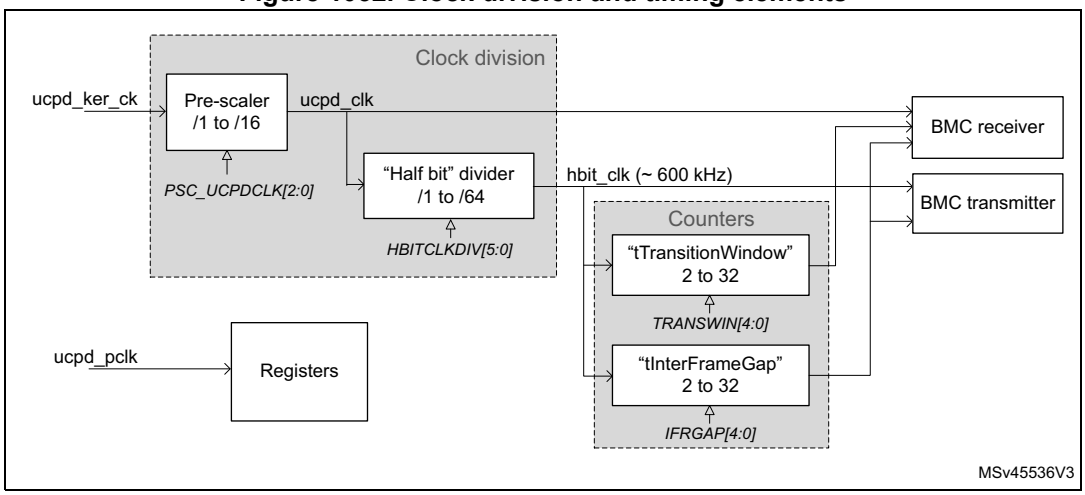
The register section is clocked with the APB clock (ucpd_pclk).

The main functional part of the transmitter is clocked with ucpd_clk clock, pre-scaled from the ucpd_ker_ck clock according to the PSC_UCPDCLK[2:0] bitfield of the UCPD_CFGR1 register. The main functional part of the receiver is clocked with the ucpd_rx_clk recovered from the incoming bitstream.

The receiver is designed to work in the clock frequency range from 6 to 18 MHz. However, the optimum performance is ensured in the range from 6 to 12 MHz.

The following diagram shows the clocking and timing elements of the UCPD peripheral.

Figure 1032. Clock division and timing elements



Refer to the USB PD specification in order to set appropriate delays. For *tTransitionWindow* and especially for *tInterFrameGap*, the clock frequency uncertainty must be taken into account so as to respect specified timings in all cases.

51.4.3 Physical layer protocol

The physical layer covers the signaling underlying the USB Power Delivery specification.

On the transmitter side its main function is to form packets according to the defined packet format including generally:

- preamble
- start of packet (SOP, ordered set)
- payload header
- payload data
- cyclic redundancy check (CRC) information
- end of packet (EOP)

Before going on the CC line, the data stream is BMC-encoded, respecting specified timing restrictions.

On the receive side, the principle task is to:

- extract start of packet (SOP, ordered set) information
- extract payload header
- extract payload data
- receive and check CRC
- receive end of packet (EOP)

The receive is basically a reverse of the transmit process, thus starting with BMC data stream decoding.

Symbol encoding

Apart from the preamble all symbols are encoded with a 4b5b scheme according to the specification shown in the following table.

Table 595. 4b5b symbol encoding table

Name	4b	5b	Symbol description
0	0000	11110	hex data 0
1	0001	01001	hex data 1
2	0010	10100	hex data 2
3	0011	10101	hex data 3
4	0100	01010	hex data 4
5	0101	01011	hex data 5
6	0110	01110	hex data 6
7	0111	01111	hex data 7
8	1000	10010	hex data 8
9	1001	10011	hex data 9
A	1010	10110	hex data A
B	1011	10111	hex data B
C	1100	11010	hex data C

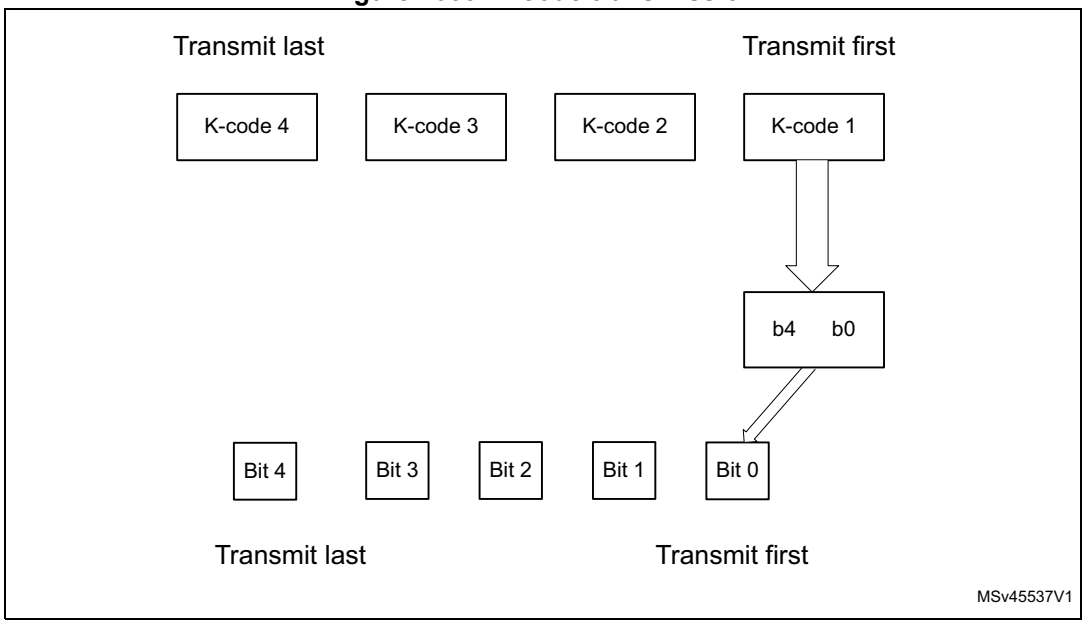
Table 595. 4b5b symbol encoding table (continued)

Name	4b	5b	Symbol description
D	1101	11011	hex data D
E	1110	11100	hex data E
F	1111	11101	hex data F
Sync-1	K-code	11000	Startsynch #1
Sync-2	K-code	10001	Startsynch #2
RST-1	K-code	00111	Hard Reset #1
RST-2	K-code	11001	Hard Reset #2
EOP	K-code	01101	EOP
Reserved	Error	00000	Do Not Use
Reserved	Error	00001	Do Not Use
Reserved	Error	00010	Do Not Use
Reserved	Error	00011	Do Not Use
Reserved	Error	00100	Do Not Use
Reserved	Error	00101	Do Not Use
Sync-3	K-code	00110	Startsynch #3
Reserved	Error	01000	Do Not Use
Reserved	Error	01100	Do Not Use
Reserved	Error	10000	Do Not Use
Reserved	Error	11111	Do Not Use

Ordered sets

An ordered set consists of four K-codes as shown in the following figure.

Figure 1033. K-code transmission



The following table lists the defined ordered sets, including all possible SOP* sequences.

At the physical layer, the Hard Reset has higher priority than the other ordered sets so it can interrupt an ongoing Tx message.

Table 596. Ordered sets

Ordered set name	K-code #1	K-code #2	K-code #3	K-code #4
SOP	Sync-1	Sync-1	Sync-1	Sync-2
SOP’	Sync-1	Sync-1	Sync-3	Sync-3
SOP’’	Sync-1	Sync-3	Sync-1	Sync-3
Hard Reset	RST-1	RST-1	RST-1	RST-2
Cable Reset	RST-1	Sync-1	RST-1	Sync-3
SOP’_Debug	Sync-1	RST-2	RST-2	Sync-3
SOP’’_Debug	Sync-1	RST-2	Sync-3	Sync-2

On reception, the physical layer must accept ordered sets with any combination of three correct K-codes out of four, as shown in the following table:

Table 597. Validation of ordered sets

Status	1st code	2nd code	3rd code	4th code
Valid	Corrupt	K-code	K-code	K-code
Valid	K-code	Corrupt	K-code	K-code

Table 597. Validation of ordered sets (continued)

Status	1st code	2nd code	3rd code	4th code
Valid	K-code	K-code	Corrupt	K-code
Valid	K-code	K-code	K-code	Corrupt
Valid (perfect)	K-code	K-code	K-code	K-code
Not valid (example)	K-code	Corrupt	K-code	Corrupt

Bit ordering at transmission

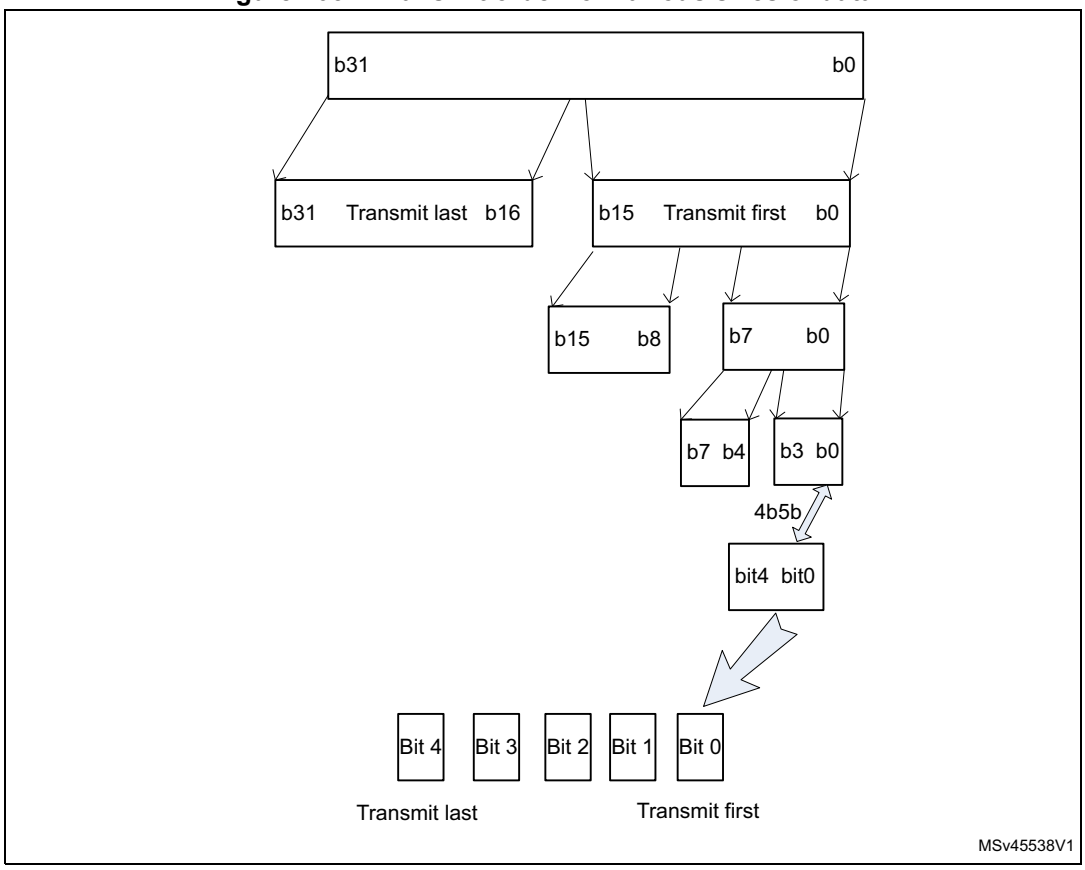
Allowed transmission data units / data sizes are in the following table.

Table 598. Data size

Data unit	Non-encoded	Encoded
Byte	8-bits	10-bits
Word	16-bits	20-bits
DWord	32-bits	40-bits

The bit transmission order is shown in the following figure.

Figure 1034. Transmit order for various sizes of data

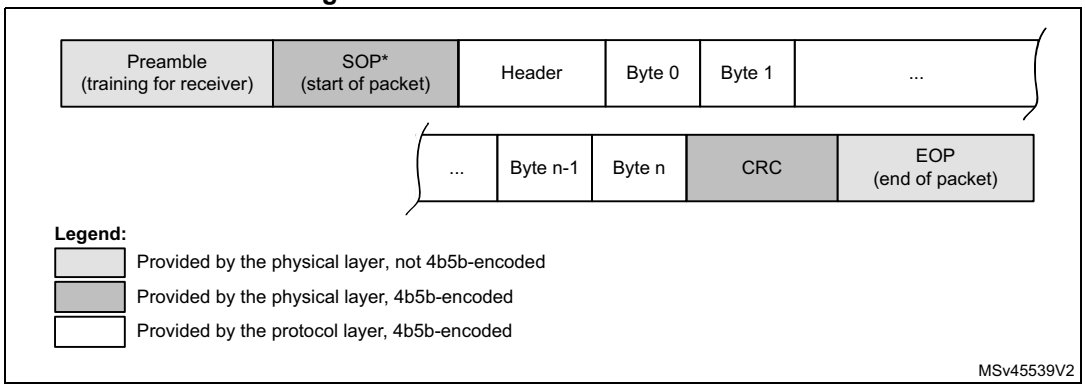


Packet format

Messages other than Hard Reset and Cable Reset

The packet format is shown in the following figure, with information on 4b5b encode and data source.

Figure 1035. Packet format



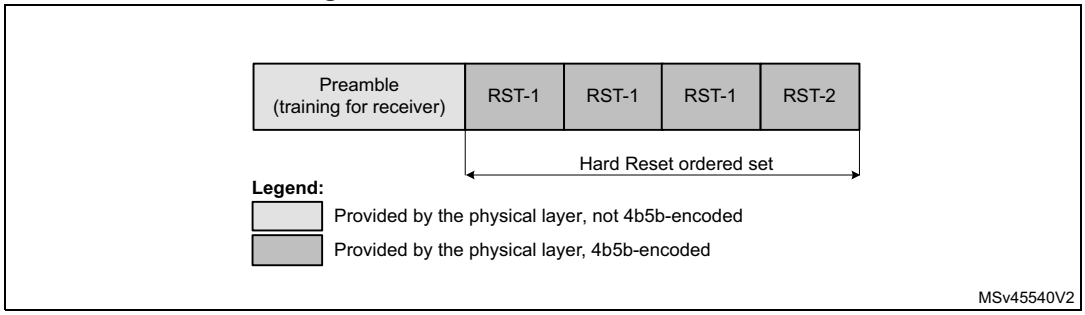
Hard Reset

The physical layer handles the Hard Reset signaling differently than the other types of message as it has higher priority to be able to interrupt an ongoing transfer.

The physical layer specification implies the following sequence in the case of an ongoing Tx message:

1. Terminate the message by sending an EOP K-code and discard the rest of the message.
2. Wait for *tInterFrameGap* time.
3. If the CC line is not idle, wait until it goes idle.
4. Send the preamble followed by the four K-codes of Hard Reset signaling.
5. Disable the CC channel (stop sending and receiving), reset the physical layer and inform the protocol layer that the physical layer is reset.
6. Re-enable the channel when requested by the protocol layer.

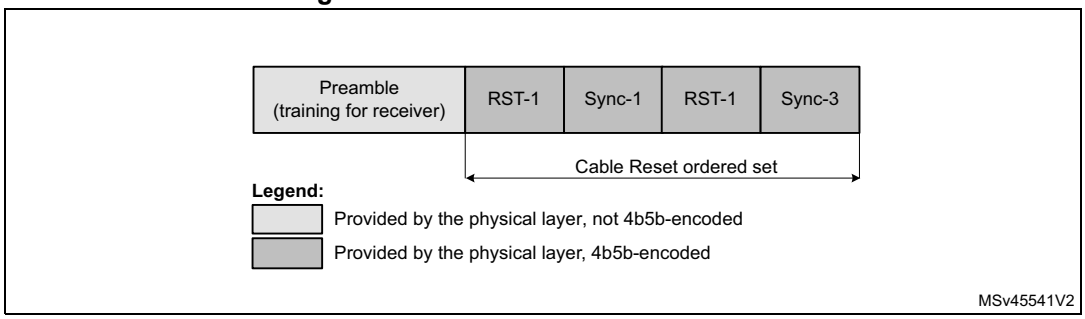
Figure 1036. Line format of Hard Reset



Cable Reset

Cable Reset shown in the following figure is similar in format to Hard Reset, but unlike Hard Reset it does not require a specific high-priority treatment.

Figure 1037. Line format of Cable Reset



Collision avoidance

The physical layer respects the *tInterFrameGap* delay between end of last-transmitted bit of a Tx message, and the first bit of a following message.

It also checks the idle state of the CC line before starting transmission. The CC line is considered idle if it shows less than three (*nTransitionCount*) transitions within *tTransitionWindow* (12 to 20 µs). The Power Delivery specification revision 3.1 also requires to manage the Rp value (source) and monitor Type-C voltage level for these Rp modifications (at the sink).

Physical layer signaling schemes

The bit are signaled with bi-phase mark coding (BMC).

BIST

Depending on the BIST action required by the protocol layer, either of the following can be run:

- a Tx BIST pattern test, achieved by writing TXMODE and TXSEND
- an Rx BIST pattern test, achieved by writing RXMODE to the correct value for RXBIST.

The two possible patterns supported in UCPD (corresponding to “BMC” mode) are:

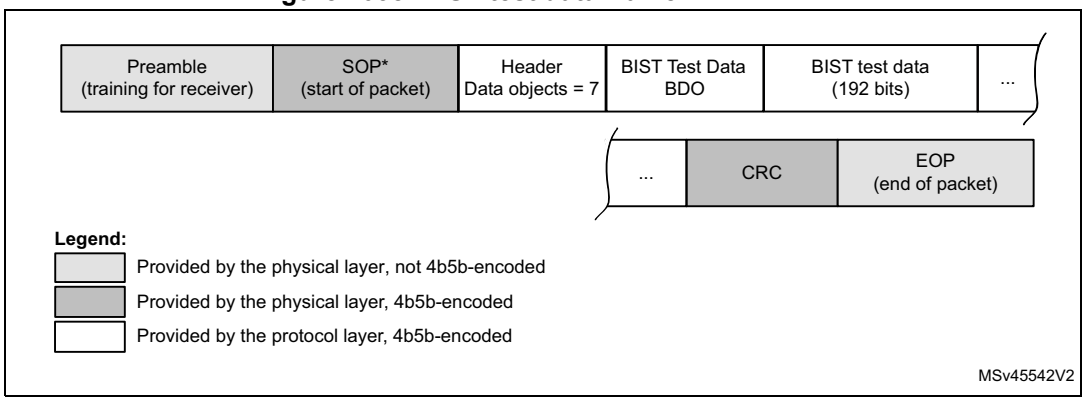
- BIST Test Data (192 bit pattern), applies to Tx and Rx. In the case of Rx, the message is received (but discarded rather than passing to the protocol layer, which must nevertheless still generate a GoodCRC Tx message in acknowledgment).
- BIST Carrier Mode 2 (single pattern, infinite length message), applies to Tx only. As opposed to Tx, the receiver in this mode simply ignores the CC line during this state.

BIST test data pattern

The test data pattern is not viewed as a special case in UCPD.

The BIST test data packet frame format is shown in the following figure.

Figure 1038. BIST test data frame



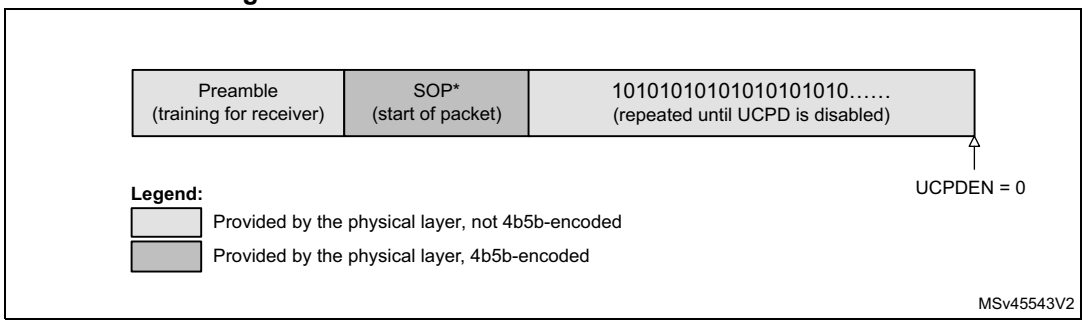
This is a fixed length test data pattern. In reality the only aspect that marks its difference from the general packet format already shown in *Figure 1035: Packet format* is the contents of the Header. As UCPD receives the Tx Header contents via programming (it is simply viewed as part of the payload), it is only this programming (and not the block's behavior) that differentiates the general packet from the BIST Test Data packet.

BIST Carrier Mode 2

When required, this BIST test mode sends an alternating pattern of 1010 that is continually repeated. As this mode is intended for signal analysis it is stable condition with (in V1.0 of the USB PD specification) no defined length. Starting from V1.1 of the USB PD specification, the protocol layer defines a counter that indicates when to exit this mode.

The way to quit the infinite 1010 sequence (according to requirements of the USB PD specification) is to disable the UCPD peripheral via the UCPDEN bit.

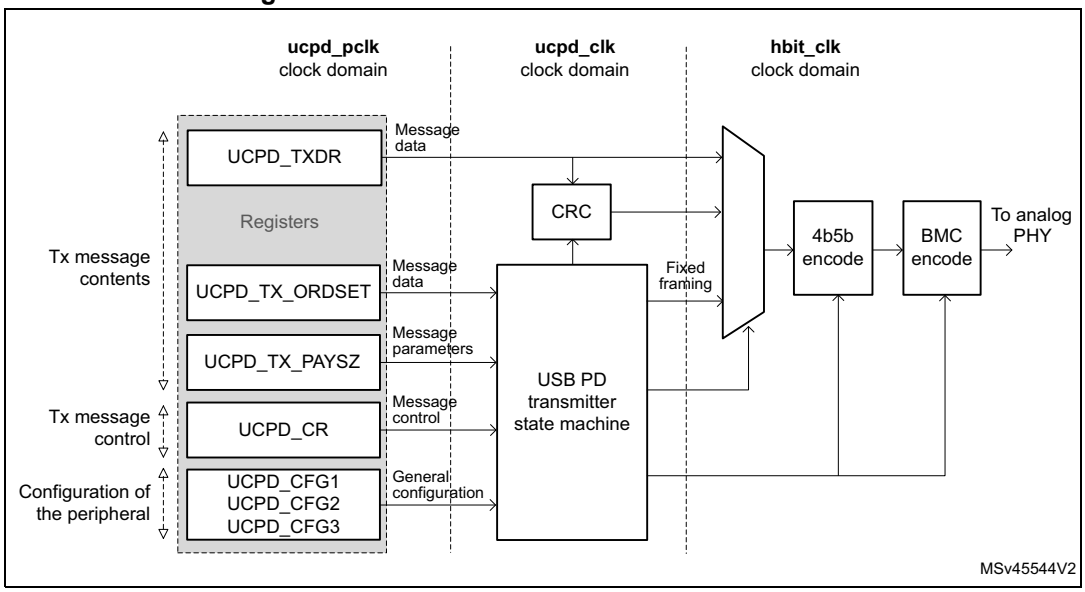
Figure 1039. BIST Carrier Mode 2 frame



51.4.4 UCPD BMC transmitter

The BMC transmitter comprises 4b5b encoding, CRC generation, and BMC encode, as shown in the following figure. Its output goes to the analog PHY through a channel switch.

Figure 1040. UCPD BMC transmitter architecture



BMC encoder

The bi-phase mark coding method is defined in the *IEC 60958-1 Digital Audio Interface Part:1 General Edition 3.0 2008-09 www.iec.ch* specification.

The half-bit clock hbit_clk is derived from ucpd_clk through a simple divider controlled by the HBITCLKDIV[5:0] bitfield of the UCPD_CFGR1 register. This ensures the same duration of high and low half-bit periods (if neglecting a small difference due to different rising and falling edge duration and due to jitter), and the same bit duration (if neglecting jitter).

Transmitter timing and collision avoidance

Hardware support of collision avoidance is made as a function of the half bit time for the transmitter. Two counters are implemented:

– *tInterFrameGap*: via IFRGAP (pre-defined value, can be altered)
– *tTransitionWindow*: via TRANSWIN (pre-defined value, can be altered)

These two counters once set correctly generates the interframe gap.

Hard Reset in transmitter

In order to facilitate generation of a Hard Reset, a special code of TXMODE field is used. No other fields need to be written.

On writing the correct code, the hardware forces Hard Reset Tx under the correct (optimal) timings with respect to an ongoing Tx message, which (if still in progress) is cleanly terminated by truncating the current sequence and directly appending an EOP K-code sequence. No specific interrupt is generated relating to this truncation event.

Transmitter behavior in the case of errors

The under-run condition (TXUND interrupt) may happen by accident and in this case, the UCPD is starved of (the correct) Tx payload and is not able to complete the Tx message correctly. This is a serious error (for this to happen the software fails to respond in time). As a result the hardware ensures the CRC is incorrect at the end of the message, thus guaranteeing the message to be discarded at the receiver.

51.4.5 UCPD BMC receiver

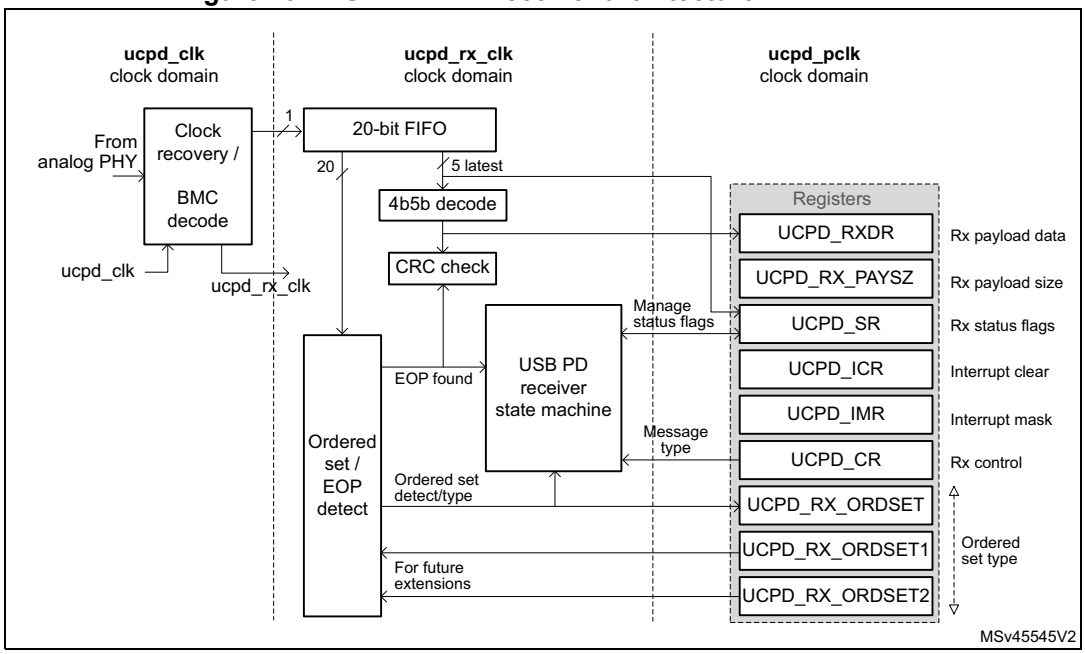
The UCPD BMC receiver performs:

- Clock recovery
- Preamble detection / timing derivation
- BMC decoding
- 4b5b decoding
- K-code ordered set recognition
- CRC checking
- SOP detection
- EOP detection

The receiver is activated as soon as the UCPD peripheral is enabled (via UCPDEN), but it waits for an idle CC line state before attempting to receive a message.

The following figure shows the UCPD BMC receiver high-level architecture.

Figure 1041. UCPD BMC receiver architecture



CRC checker

The received bits are fed into a CRC checker which evolves a 32-bit state during the received the payload bitstream. At the end the 32 bits of the CRC also fed into the logic

The EOP detection (5 bits) halts the process and a check is performed for the fixed residual state which confirms correct reception of the payload (in fact the residual is 0xC704DD78).

At this point the UCPD raises interrupt RXMSGEND. If the CRC was not correct then RXERR is set true and the receive data must be discarded.

Under normal operation, this interrupt would previously have been acknowledged and thus cleared. If this is not the case, a different interrupt RXOVR is generated in place of RXMSGEND.

Ordered set detection

This function detects the different ordered sets each consisting of four 5-bit K-codes.

Once we are in the preamble we opens a sliding window detection of the ordered set (4 words of 5 bits).

The ordered sets detected include all SOP* codes (SOP, SOP', and SOP''), but also Hard Reset, Cable Reset, SOP'_Debug, SOP''_Debug, and two extensions defined by registers UCPD_RX_ORDEXT1 and UCPD_RX_ORDEXT2.

EOP detection and Hard Reset exception handling

EOP is a fixed 5-bit K-code marking the end of a message.

The way in which a transmitter is required to send a Hard Reset (if a previous message transmit is still in progress) is that this previous message is truncated early with an EOP.

If Hard Reset were ignored, then the EOP detection can be done only at the expected time. However, due to the Hard Reset issue, the EOP detector must be active while an Rx message is arriving. When an "early EOP" is detected, the truncated Rx message is immediately discarded.

Truncated or corrupted message exception

Once the ordered set has been detected, depending on the message, there may be data bytes to be received which is completed with a CRC and EOP. If at any point during these phases an error condition happens:

- the line becomes static for a time significantly longer than one "UI" period (the exact threshold for this condition is not critical but the exception must occur before three UIs), or
- the message goes to the end but it is not recognized (for example EOP is corrupted).

In both cases, the receiver quits the current message, raising RXMSGEND and RXERR flags.

Short preamble or incomplete ordered set exception

In the exceptional case of the receiver seeing less that half of the expected preamble, the frequency estimation allowing correct BMC-decode becomes impossible. Even if the full preamble is seen, allowing frequency estimation, but the ordered set is not fully received before the line becomes static, the receiver state machine does not start.

In both of these cases, the clock-recovery/BMC decoder re-starts, checking initially for an IDLE condition, followed by a preamble, and then estimating frequency.

51.4.6 UCPD Type-C pull-ups (Rp) and pull-downs (Rd)

UCPD offers simple control of these resistors via ANAMODE and ANASUBMODE[1:0]. In case only one of the CC lines is to be used, it is possible to optimize power consumption by disabling control on the other line, through the CCENABLE[1:0] bitfield.

When the MCU is unpowered, it still presents the "dead battery" Rd, provided that UCPDx_DBCC1 and UCPDx_DBCC2 pins are each connected to UCPDx_CC1 and UCPDx_CC2 pins, respectively.

If dead battery behavior is not required (for example for source only products), then UCPDx_DBCC1 and UCPDx_DBCC2 pins must both be tied to ground.

After power arrives and the MCU boots, the desired behavior (for example source) must be programmed into ANAMODE and ANASUBMODE[1:0] before setting the UCPD_DBDIS bit of the PWR_UCPDR register to remove dead battery pull-down resistor and allow the values just programmed to take effect.

Use of Standby low-power mode is possible for sinks in the unattached state.

51.4.7 UCPD Type-C voltage monitoring and de-bouncing

For correct operation of the Type-C state machine and for detecting the cable orientation, the CC1/2 lines must be monitored for voltage level, while ignoring fast events such as peaks.

Thresholds between voltage levels on the CC1/2 lines are determined through PHY threshold detector settings.

The TYPEC_VSTATE_CC1/2[1:0] bitfields reflect the CC1/2 line levels processed with a hardware de-bouncing filter that suppresses high-speed line events such as peaks. The PHYCCSEL bit selects the line, CC1 or CC2, to be used for Power Delivery signaling.

For minimizing the power consumption, it is recommended to use the polling method, with the Type-C detectors only turned on for the instant of polling, rather than keeping the Type-C detectors permanently on and wake the device up from Stop mode upon CC1/2 line events.

51.4.8 UCPD fast role swap (FRS)

FRS signaling

The FRS condition (a pulse of a specific length), is generated upon setting the FRSTX bit.

For the duration of FRS condition, the currently active I/O configured as UCPD_FRSTX1 (or 2) (alternate function) controls, with high level, the gate of an external NMOS transistor that pulls the active CC line down.

FRS detection

FRS monitoring is enabled by setting the bit FRSRXEN, after writing PHYCCSEL that selects the active CC line depending on the cable orientation detected.

51.4.9 UCPD DMA Interface

DMA is implemented in the UCPD and when it is enabled the byte-level interrupts to handle UCPD1_TXDR and UCPD1_RXDR registers (Tx and Rx data register, each one byte) are no longer needed.

By enabling bits TXDMAEN and/or RXDMAEN, DMA can be activated independently for Tx and/or Rx functionality.

51.4.10 Wake-up from Stop mode

For power consumption optimization, it is useful to use Stop mode and wait for events on CC lines to wake the MCU up.

In order for this to work, it must be first enabled by writing a 1 to WUPEN.

The events causing the wake-up can be:

- Events on the BMC receiver (RXORDDET, RXHRSTDET), hardware enable PHYRXEN
- Event on the FRS detector (FRSEVT), hardware enable FRSRXEN
- Events on the Type-C detectors (TYPECEVT1, TYPECEVT2), hardware enables CC1TCDIS, CC2TCDIS

51.5 UCPD programming sequences

The normal sequence of use of the UCPD unit is:

1. Configure UCPD.
2. Enable UCPD.
3. Concurrently:
   – On demand from protocol layer, send Tx message
   – Intercept (poll or wait for interrupt) relevant Rx messages and recover detail to hand off to protocol layer

Repeat the last point infinitely.

51.5.1 Initialization phase

Use the following sequence for a clean startup:

1. Prepare all initial clock divider values, by writing the UCPD_CFG register.
2. Enable the unit, by setting the UCPDEN bit.
3. Enable the analog Rx filter of either CC line, via the RXAFILTEN bit of the UCPD_CFGR2 register.

51.5.2 Type-C state machine handling

For the general application cases of source, sink, or dual-role port (the last alternating the source and the sink), the software must implement a corresponding USB Type-C state machine. The basic coding is in the following table.

Table 599. Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx

ANAMODE	ANASUBMODE[1:0]	Notes	TYPEC_VSTATE_CCx[1:0]			
			00	01	10	11
0: Source	00: Disabled	Disabled	N/A			
	01: Default USB Rp	-	vRa[Def]	vRd[Def]	vOPEN[Def]	N/A
	10: 1.5A Rp	-	vRa[1.5]	vRd[1.5]	vOPEN[1.5]	
	11: 3.0A Rp	-	vRa[3.0]	vRd[3.0]	vOPEN[3.0]	
1: Sink	xx	-	vRa	vRd-USB	vRd-1.5	vRd-3.0

The CCENABLE[1:0] bitfield can disable pull-up/pull-downs on one of the CC lines.

Note: The Type-C state machine depends not only on CC line levels, but also on VBUS presence detection (sink mode) and, when in source mode, determines VCONN generation and

VBUS state (ON/OFF/+voltage level); discharge). UCPD does not directly control VBUS generation circuitry nor VCONN load switch (enabling VCONN supply generator to be connected to the CC line), but the application needs these inputs and controls, to function correctly.

General programming sequence (with UCPD configured then enabled)

1. Set ANAMODE and ANASUBMODE[1:0] based on the current position in USB Type-C state machine (and also the current advertisement in the case of a source). This turns on the appropriate pull-ups/pull-downs on the CC lines, and defines the voltage levels that the TYPEC_VSTATE fields represent. Note that before programming, the PHY is effectively off.
2. Read TYPEC_VSTATE_CC1/2 to determine the initial Type-C state (for example whether the local source is connected to a remote sink).
3. In the case of no connection, wait for a connection event.
4. Assuming a connection is detected and assuming a local Power Delivery functionality is implemented, start sending/receiving Power Delivery messages.
5. When a new interrupt/event occurs on PHYEVT1/2 indicating a change in stable voltage, re-evaluate the implications and give this input to the Type-C state machine.

Case of a source that needs to change between one of the three possible Rp values (Default-USB / 1.5A / 3.0A) and the sink connected to it:

- [Source] Simply reprogram ANASUBMODE[1:0]
- [Sink behavior from that time] PHYEVT1/2 occurs and the TYPEC_VSTATE1/2 changes accordingly

Programming for a dual-role port (DRP) toggling from source to sink:

- Simply re-program ANAMODE and ANASUBMODE[1:0] to start the new behavior

Detailed programming sequence (example):

Table 600. Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2)

Type-C state	ANAMODE; ANASUBMODE[1:0]	CCENABLE	PHYCCSEL	RDCH	CC[x] VCONNEN(1)	Event => go to next line	Comments
Unattached. SRC	0:Source; 11:Rp3A0	11:both enabled	0 (don't care)	0: [Normal]	00: [neither]	PHYEVT 1: [VRd-3A0]	Wait for sink attach detect ; seen on CC1 [EVT1]
Attachwait. SRC			0 [Rd on CC1]			PHYEVT 2: [VRa]	Attachwait started (100-200 ms) ; now also see the Ra => requesting VCONN
Attached. SRC [VCONN => CC2]	0:Source; 11:Rp3A0 [SinkTxOK]	01: CC2 disable (possible and recommended due to external VCONN switch)			10: [CC2 active]	Timer (100 ms) and no PHYEVTx	Local CC2 disconnected from PHY (VCONN switch connects VCONN source to CC2 externally; Continue to monitor PHYEVT1
	0:Source; 10:Rp1A5 [SinkTxNG]					SW timers (SinkTxNG)	Source wants to initiate message sequence (SinkTxNG condition set first)
	0:Source; 11:Rp3A0 [SinkTxOK]						Source finished message sequence (SinkTxOK condition afterwards)
						PHYEVT 1: [VOpen-3A0]	Wait for Sink disconnected (Vopen on CC1)
Unattached wait. SRC		11:both enabled	0 (do not care)	1: [discharge]	00: [neither]	> 0.8V detection	Discharge VCONN [CC2] actively [Rdch]; to < 0.8V
Unattached. SRC	0:Source; 11:Rp3A0			0: [Normal]			[Details as first line of table]

1. Two GPIOs to enable VCONN through external load switch components

51.5.3 USB PD transmit

On reception of a message from the protocol layer (that is, to be sent), prepare Tx message contents by writing the UCPD_TX_ORDSET and UCPD_TX_PAYSZ registers.

The message transmission is triggered by setting the TXSEND bit, with an appropriate value of the TXMODE bitfield.

When the data byte is transmitted, the TXIS flag is raised to request a new data write to the UCPD_TXDR register.

This re-iterates until the entire payload of data is transmitted.

Upon sending the CRC packet, the TXMSGSENT flag is set to indicate the completion of the message transmission.

Hard Reset transmission

As soon as it is known that a Hard Reset needs to be transmitted, setting the TXHRST bit of the UCPD_CR register forces the internal state machine to generate the correct sequence. The value of UCPD_TX_ORDSET does not require update in this precise case (the correct code for Hard Reset is sent by UCPD).

The USB Power Delivery specification requires that in the case of an ongoing message transmission, the Hard Reset takes precedence. In this case, for example, UCPD truncates the payload of the current message, appending EOP to the end. No notification is available via the registers (for example through the TXMSGSEND flag). This is justified by the fact that the Hard Reset takes precedence over any previous activity (for which it is therefore no longer important to know if it is completed).

Use of DMA for transmission

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the UCPD_CR register.

For each message:

- Prepare the whole message in memory (starting with two header bytes)
- Program the DMA operation with a length corresponding to the two header bytes plus a number of data bytes corresponding to the number of data words multiplied by four
- Write TXSEND to initiate the message transfer
- If TXMSGDISC then go back to previous line (TXSEND)
- Wait for DMA transfer complete interrupt (that is, when last Tx byte written to UCPD)
- Double-check subsequent TXMSGSENT interrupt appears

51.5.4 USB PD receive

Notification of start of the receive message sequence is triggered by an interrupt on UCPD_SR (bit RXORDDET).

The information is recovered by reading:

- UCPD_RX_SOP (on interrupt RXORDDET)
- UCPD_RXDR (on interrupt RXNE, repeats for each byte)
- UCPD_RXPAYSZ (on interrupt RXMSGEND)

The data previously read from UCPD_RXDR above must be discarded at this point if the RXERR flag is set.

If the CRC is valid, the received data is transferred to the protocol layer.

For debug purposes, it may be desirable to track statistics of the number of incorrect K-codes received (this is done only when 3/4 K-codes were valid as defined in the specification). This is facilitated through:

- RXSOP3OF4 bit indicating the presence of at least one invalid K-code
- RXSOPKINVALID bitfield identifying the order of invalid K-code in the ordered set

Use of DMA for reception

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the UCPD_CR register.

Whenever a Rx message is expected:

- Program a DMA receive operation (and spare buffer) a little longer than the maximum possible message (length depends on extended message support).
- After receiving RXORDDET, DMA operation starts working in the background.
- On reception of RXMSGEND interrupt, read RXPAYSZ.
- Double-check RXPAYSZ vs. the number of DMA Rx bytes (must correspond but DMA read of RXDR is slightly after RXDR gets last byte).
- Process the DMA Rx buffer.
- Prepare next Rx DMA buffer as soon as possible in order to be ready.

51.6 UCPD low-power modes

A summary of low-power modes is shown below in *Table 601: Effect of low power modes on the UCPD*.

Table 601. Effect of low power modes on the UCPD

Mode	Description
Sleep	No effect
Stop	Detection of events (Type-C, BMC Rx, FRS detection) remains operational and can wake up the MCU.
Standby	UCPD is not operating, and cannot wake up the MCU. Pull-downs remain active if configured.
Unpowered	Dead battery pull-downs remain active.

The UCPD is able to wake up the MCU from Stop mode when it recognizes a relevant event, either:

- Type-C event relating to a change in the voltage range seen on either of the CC lines, visible in TYPEC_VSTATE_CCx
- Power delivery receive message with an ordered set matching those filtered according to RXORDSETEN[8:0], visible by reading RXORDSET

Wake-up from Stop mode is enabled by setting the WUPEN bit in the UCPD_CFG2 register.

At UCPD level three types of event requiring kernel clock activity may occur during Stop mode:

- Activity on the analog PHY voltage threshold detectors which can later be confirmed to be a stable change between voltage ranges defined in the Type-C specification
- Activity on Power Delivery BMC receiver (coming from the selected CC line) which can potentially generate an Rx message event (that is, RXORDSET) later
- Activity on Power Delivery FRS detector which can potentially generate an FRS signaling detection event (that is, FRSEVT) later

It order to function correctly with the RCC, the clock request signal is activated (conditional on WUPEN) when there is asynchronous activity on:

- Type-C voltage threshold detectors (coming from either CC line)
- Power Delivery receiver signal (from the selected CC line)
- FRS detection signal (from the selected CC line)

51.7 UCPD interrupts

The table below lists the UCPD event flags, with the associated flag clear bits and interrupt enable bits.

Table 602. UCPD interrupt requests

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
FRS detection	FRSEVT	Set FRSEVTCF	FRSEVTIE
Type C voltage level change on CC2	TYPECEVT2	Set TYPECEVT2CF	TYPECEVT2IE
Type C voltage level change on CC1	TYPECEVT1	Set TYPECEVT1CF	TYPECEVT1IE
Rx message received	RXMSGEND	Set RXMSGENDCF	RXMSGENDIE
Rx data overflow	RXOVR	Set RXOVRCF	RXOVR
Rx Hard Reset detected	RXHRSTDET	Set RXHRSTDETCF	RXHRSTDETIE
Rx ordered set (4 K-codes) detected	RXORDDET	Set RXORDDETCF	RXORDDETIE
Receive data register not empty	RXNE	Read data in UCPD_RXDR	RXNEIE
Tx data underrun	TXUND	Set TXUNDCF	TXUNDIE
Hard Reset sent	HRSTSENT	Set HRSTSENTCF	HRSTSENTIE
Hard Reset discarded	HRSTDISC	Set HRSTDISCCF	HRSTDISCIE
Transmit message aborted	TXMSGABT	Set TXMSGABTCF	TXMSGABTIE
Transmit message sent	TXMSGSENT	Set TXMSGSENTCF	TXMSGSENTIE
Transmit message discarded	TXMSGDISC	Set TXMSGDISCCF	TXMSGDISCIE
Transmit data required	TXIS	Write data to the UCPD_TXDR register	TXISIE

When an interrupt from the UCPD is received, then the software has to check what is the source of the interrupt by reading the UCPD_SR register.

Depending on which bit is at 1, the ISR must handle that condition and clear the bit by a write to the appropriate bit of the UCPD_ICR register.

51.8 UCPD registers

51.8.1 UCPD configuration register 1 (UCPD_CFGR1)

Address offset: 0x000

Reset value: 0x0000 0000

General configuration of the peripheral. Writing to this register is only effective when UCPD is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UCPDEN	RXDMAEN	TXDMAEN	RXORDSETEN[8:0]									PSC_UCPDCLK[2:0]			Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRANSWIN[4:0]					IFRGAP[4:0]					HBITCLKDIV[5:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UCPDEN**: UCPD peripheral enable

General enable of the UCPD peripheral.

0: Disable

1: Enable

Upon disabling, the peripheral instantly quits any ongoing activity and all control bits and bitfields default to their reset values. They must be set to their desired values each time the peripheral transits from disabled to enabled state.

Bit 30 **RXDMAEN**: Reception DMA mode enable

When set, the bit enables DMA mode for reception.

0: Disable

1: Enable

Bit 29 **TXDMAEN**: Transmission DMA mode enable

When set, the bit enables DMA mode for transmission.

0: Disable

1: Enable

Bits 28:20 **RXORDSETEN[8:0]**: Receiver ordered set enable

The bitfield determines the types of ordered sets that the receiver must detect. When set/cleared, each bit enables/disables a specific function:

0bXXXXXXXX1: SOP detect enabled

0bXXXXXXX1X: SOP' detect enabled

0bXXXXXX1XX: SOP'' detect enabled

0bXXXXX1XXX: Hard Reset detect enabled

0bXXXX1XXXX: Cable Detect reset enabled

0bXXX1XXXXX: SOP'_Debug enabled

0bXX1XXXXXX: SOP''_Debug enabled

0bX1XXXXXXX: SOP extension#1 enabled

0b1XXXXXXXX: SOP extension#2 enabled

Bits 19:17 **PSC_UCPDCLK[2:0]**: Pre-scaler division ratio for generating ucpd_clk

The bitfield determines the division ratio of a kernel clock pre-scaler producing UCPD peripheral clock (ucpd_clk).

0x0: 1 (bypass)
0x1: 2
0x2: 4
0x3: 8
0x4: 16

It is recommended to use the pre-scaler so as to set the ucpd_clk frequency in the range from 6 to 9 MHz.

Bit 16 Reserved, must be kept at reset value.

Bits 15:11 **TRANSWIN[4:0]**: Transition window duration

The bitfield determines the division ratio (the bitfield value minus one) of a hbit_clk divider producing *tTransitionWindow* interval.

0x00: Not supported
0x01: 2
0x09: 10 (recommended)
0x1F: 32

Set a value that produces an interval of 12 to 20 us, taking into account the ucpd_clk frequency and the HBITCLKDIV[5:0] bitfield setting.

Bits 10:6 **IFRGAP[4:0]**: Division ratio for producing inter-frame gap timer clock

The bitfield determines the division ratio (the bitfield value minus one) of a ucpd_clk divider producing inter-frame gap timer clock (*tInterFrameGap*).

0x00: Not supported
0x01: 2
0x0D: 14
0x0E: 15
0x0F: 16
0x1F: 32

The division ratio 15 is to apply for Tx clock at the USB PD 2.0 specification nominal value. The division ratios below 15 are to apply for Tx clock below nominal, and the division ratios above 15 for Tx clock above nominal.

Bits 5:0 **HBITCLKDIV[5:0]**: Division ratio for producing half-bit clock

The bitfield determines the division ratio (the bitfield value plus one) of a ucpd_clk divider producing half-bit clock (hbit_clk).

0x00: 1 (bypass)
0x1A: 27
0x3F: 64

51.8.2 UCPD configuration register 2 (UCPD_CFGR2)

Address offset: 0x004

Reset value: 0x0000 0000

Configuration of the UCPD Rx signal filtering. Writing to this register is only effective when UCPD is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXAFILTEN	Res.	Res.	Res.	Res.	WUPEN	FORCECLK	RXFILT2N3	RXFILTDIS
							rw					rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **RXAFILTEN**: Rx analog filter enable
Setting the bit enables the Rx analog filter required for optimum Power Delivery reception.
0: Disable
1: Enable

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **WUPEN**: Wake-up from Stop mode enable
Setting the bit enables the UCPD_ASYNC_INT signal.
0: Disable
1: Enable

Bit 2 **FORCECLK**: Force ClkReq clock request
0: Do not force clock request
1: Force clock request

Bit 1 **RXFILT2N3**: BMC decoder Rx pre-filter sampling method
Number of consistent consecutive samples before confirming a new value.
0: 3 samples
1: 2 samples

Bit 0 **RXFILTDIS**: BMC decoder Rx pre-filter enable
0: Enable
1: Disable
The sampling clock is that of the receiver (that is, after pre-scaler).

51.8.3 UCPD control register (UCPD_CR)

Address offset: 0x00C

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2TCDIS	CC1TCDIS	Res.	RDCH	FRSTX	FRSRXEN
										rw	rw		rw	rs	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CCENABLE[1:0]		ANAMODE	ANASUBMODE[1:0]		PHYCCSEL	PHYRXEN	RXMODE	TXHRST	TXSEND	TXMODE[1:0]	
				rw	rw	rw	rw	rw	rw	rw	rw	rs	rs	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CC2TCDIS**: CC2 Type-C detector disable

The bit disables the Type-C detector on the CC2 line.

0: Enable

1: Disable

When enabled, the Type-C detector for CC2 is configured through ANAMODE and ANASUBMODE[1:0].

Bit 20 **CC1TCDIS**: CC1 Type-C detector disable

The bit disables the Type-C detector on the CC1 line.

0: Enable

1: Disable

When enabled, the Type-C detector for CC1 is configured through ANAMODE and ANASUBMODE[1:0].

Bit 19 Reserved, must be kept at reset value.

Bit 18 **RDCH**: Rdch condition drive

The bit drives Rdch condition on the CC line selected through the PHYCCSEL bit (thus associated with VCONN), by remaining set during the source-only *UnattachedWait.SRC* state, to respect the Type-C state. Refer to *"USB Type-C ECN for Source VCONN Discharge"*. The CCENABLE[1:0] bitfield must be set accordingly, too.

0: No effect

1: Rdch condition drive

Bit 17 **FRSTX**: FRS Tx signaling enable.

Setting the bit enables FRS Tx signaling.

0: No effect

1: Enable

The bit is cleared by hardware after a delay respecting the USB Power Delivery specification Revision 3.1.

Bit 16 **FRSRXEN**: FRS event detection enable

Setting the bit enables FRS Rx event (FRSEVT) detection on the CC line selected through the PHYCCSEL bit. 0: Disable

1: Enable

Clear the bit when the device is attached to an FRS-incapable source/sink.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:10 **CCENABLE[1:0]**: CC line enable

This bitfield enables CC1 and CC2 line analog PHYs (pull-ups and pull-downs) according to ANAMODE and ANASUBMODE[1:0] setting.
0x0: Disable both PHYs
0x1: Enable CC1 PHY
0x2: Enable CC2 PHY
0x3: Enable CC1 and CC2 PHY
A single line PHY can be enabled when, for example, the other line is driven by VCONN via an external VCONN switch. Enabling both PHYs is the normal usage for sink/source.

Bit 9 **ANAMODE**: Analog PHY operating mode

0: Source
1: Sink
The use of CC1 and CC2 depends on CCENABLE. Refer to *Table 599: Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx* for the effect of this bitfield in conjunction with ANASUBMODE[1:0].

Bits 8:7 **ANASUBMODE[1:0]**: Analog PHY sub-mode

Refer to *Table 599: Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx* for the effect of this bitfield.

Bit 6 **PHYCCSEL**: CC1/CC2 line selector for USB Power Delivery signaling

0: Use CC1 IO for Power Delivery communication
1: Use CC2 IO for Power Delivery communication
The selection depends on the cable orientation as discovered at attach.

Bit 5 **PHYRXEN**: USB Power Delivery receiver enable

0: Disable
1: Enable
Both CC1 and CC2 receivers are disabled when the bit is cleared. Only the CC receiver selected via the PHYCCSEL bit is enabled when the bit is set.

Bit 4 **RXMODE**: Receiver mode

Determines the mode of the receiver.
0: Normal receive mode
1: BIST receive mode (BIST test data mode)
When the bit is set, RXORDSET behaves normally, RXDR no longer receives bytes yet the CRC checking still proceeds as for a normal message. As this mode prevents reception of the header (containing MessageID), software has to auto-increment a received MessageID counter for inclusion in the GoodCRC acknowledge that must still be transmitted during this test.

Bit 3 **TXHRST**: Command to send a Tx Hard Reset

0: No effect
1: Start Tx Hard Reset message
The bit is cleared by hardware as soon as the message transmission begins or is discarded.

Bit 2 **TXSEND**: Command to send a Tx packet

0: No effect
1: Start Tx packet transmission
The bit is cleared by hardware as soon as the packet transmission begins or is discarded.

Bits 1:0 **TXMODE[1:0]**: Type of Tx packet

Writing the bitfield triggers the action as follows, depending on the value:
0x0: Transmission of Tx packet previously defined in other registers
0x1: Cable Reset sequence
0x2: BIST test sequence (BIST Carrier Mode 2)
Others: invalid
From V1.1 of the USB PD specification, there is a counter defined for the duration of the BIST Carrier Mode 2. To quit this mode correctly (after the "tBISTContMode" delay), disable the peripheral (UCPDEN = 0).

51.8.4 UCPD interrupt mask register (UCPD_IMR)

Address offset: 0x010

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVTIE	Res.	Res.	Res.	Res.
											r				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2IE	TYPECEVT1IE	Res.	RXMSGENDIE	RXOVRIE	RXHRSTDETIE	RXORDDETIE	RXNEIE	Res.	TXUNDIE	HRSTSENTIE	HRSTDISCIE	TXMSGABTIE	TXMSGSENTIE	TXMSGDISCIE	TXISIE
rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **FRSEVTIE**: FRSEVT interrupt enable
0: Disable
1: Enable

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **TYPECEVT2IE**: TYPECEVT2 interrupt enable
0: Disable
1: Enable

Bit 14 **TYPECEVT1IE**: TYPECEVT1 interrupt enable

Bit 13 Reserved, must be kept at reset value.

Bit 12 **RXMSGENDIE**: RXMSGEND interrupt enable
0: Disable
1: Enable

Bit 11 **RXOVRIE**: RXOVR interrupt enable
0: Disable
1: Enable

Bit 10 **RXHRSTDETIE**: RXHRSTDET interrupt enable
0: Disable
1: Enable

Bit 9 **RXORDDETIE**: RXORDDET interrupt enable
0: Disable
1: Enable

Bit 8 **RXNEIE**: RXNE interrupt enable
0: Disable
1: Enable

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TXUNDIE**: TXUND interrupt enable
0: Disable
1: Enable

Bit 5 **HRSTSENTIE**: HRSTSENT interrupt enable
0: Disable
1: Enable

Bit 4 **HRSTDISCIE**: HRSTDISC interrupt enable
0: Disable
1: Enable

Bit 3 **TXMSGABTIE**: TXMSGABT interrupt enable
0: Disable
1: Enable

Bit 2 **TXMSGSENTIE**: TXMSGSENT interrupt enable
0: Disable
1: Enable

Bit 1 **TXMSGDISCIE**: TXMSGDISC interrupt enable
0: Disable
1: Enable

Bit 0 **TXISIE**: TXIS interrupt enable
0: Disable
1: Enable

51.8.5 UCPD status register (UCPD_SR)

Address offset: 0x014

Reset value: 0x0000 0000

The flags (single-bit status bitfields) are associated with interrupt request. Interrupt is generated if enabled by the corresponding bit of the UCPD_IMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVT	TYPEC_VSTATE_CC2[1:0]		TYPEC_VSTATE_CC1[1:0]	
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2	TYPECEVT1	RXERR	RXMSGEND	RXOVR	RXHRSTDET	RXORDDET	RXNE	Res.	TXUND	HRSTSENT	HRSTDISC	TXMSGABT	TXMSGSENT	TXMSGDISC	TXIS
r	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **FRSEVT**: FRS detection event
The flag is cleared by setting the FRSEVTCF bit.
0: No new event
1: New FRS receive event occurred

Bits 19:18 **TYPEC_VSTATE_CC2[1:0]**: CC2 line voltage level
The status bitfield indicates the voltage level on the CC2 line in its steady state.
0x0: Lowest
0x1: Low
0x2: High
0x3: Highest
The voltage variation on the CC2 line during USB PD messages due to the BMC PHY modulation does not impact the bitfield value.

Bits 17:16 **TYPEC_VSTATE_CC1[1:0]**:
The status bitfield indicates the voltage level on the CC1 line in its steady state.
0x0: Lowest
0x1: Low
0x2: High
0x3: Highest
The voltage variation on the CC1 line during USB PD messages due to the BMC PHY modulation does not impact the bitfield value.

Bit 15 **TYPECEVT2**: Type-C voltage level event on CC2 line
The flag indicates a change of the TYPEC_VSTATE_CC2[1:0] bitfield value, which corresponds to a new Type-C event. It is cleared by setting the TYPECEVT2CF bit.
0: No new event
1: A new Type-C event

Bit 14 **TYPECEVT1**: Type-C voltage level event on CC1 line
The flag indicates a change of the TYPEC_VSTATE_CC1[1:0] bitfield value, which corresponds to a new Type-C event. It is cleared by setting the TYPECEVT2CF bit.
0: No new event
1: A new Type-C event

Bit 13 **RXERR**: Receive message error
The flag indicates errors of the last Rx message declared (via RXMSGEND), such as incorrect CRC or truncated message (a line becoming static before EOP is met). It is asserted whenever the RXMSGEND flag is set.
0: No error detected
1: Error(s) detected

Bit 12 **RXMSGEND**: Rx message received
The flag indicates whether a message (except Hard Reset message) has been received, regardless the CRC value. The flag is cleared by setting the RXMSGENDCF bit.
0: No new Rx message received
1: A new Rx message received
The RXERR flag set when the RXMSGEND flag goes high indicates errors in the last-received message.

Bit 11 **RXOVR**: Rx data overflow detection
The flag indicates Rx data buffer overflow. It is cleared by setting the RXOVRCF bit.
0: No overflow
1: Overflow
The buffer overflow can occur if the received data are not read fast enough.

Bit 10 **RXHRSTDET**: Rx Hard Reset receipt detection

The flag indicates the receipt of valid Hard Reset message. It is cleared by setting the RXHRSTDETCF bit.

0: Hard Reset not received

1: Hard Reset received

Bit 9 **RXORDDET**: Rx ordered set (4 K-codes) detection

The flag indicates the detection of an ordered set. The relevant information is stored in the RXORDSET[2:0] bitfield of the UCPD_RX_ORDSET register. It is cleared by setting the RXORDDETCF bit.

0: No ordered set detected

1: A new ordered set detected

Bit 8 **RXNE**: Receive data register not empty detection

The flag indicates that the UCPD_RXDR register is not empty. It is automatically cleared upon reading UCPD_RXDR.

0: Rx data register empty

1: Rx data register not empty

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TXUND**: Tx data underrun detection

The flag indicates that the Tx data register (UCPD_TXDR) was not written in time for a transmit message to execute normally. It is cleared by setting the TXUNDCF bit.

0: No Tx data underrun detected

1: Tx data underrun detected

Bit 5 **HRSTSENT**: Hard Reset message sent

The flag indicates that the Hard Reset message is sent. The flag is cleared by setting the HRSTSENTCF bit.

0: No Hard Reset message sent

1: Hard Reset message sent

Bit 4 **HRSTDISC**: Hard Reset discarded

The flag indicates that the Hard Reset message is discarded. The flag is cleared by setting the HRSTDISCCF bit.

0: No Hard Reset discarded

1: Hard Reset discarded

Bit 3 **TXMSGABT**: Transmit message abort

The flag indicates that a Tx message is aborted due to a subsequent Hard Reset message send request taking priority during transmit. It is cleared by setting the TXMSGABTCF bit.

0: No transmit message abort

1: Transmit message abort

Bit 2 **TXMSGSENT**: Message transmission completed

The flag indicates the completion of packet transmission. It is cleared by setting the TXMSGSENTCF bit.

0: No Tx message completed

1: Tx message completed

In the event of a message transmission interrupted by a Hard Reset, the flag is not raised.

Bit 1 **TXMSGDISC**: Message transmission discarded

The flag indicates that a message transmission was dropped. The flag is cleared by setting the TXMSGDISCCF bit.

0: No Tx message discarded

1: Tx message discarded

Transmission of a message can be dropped if there is a concurrent receive in progress or at excessive noise on the line. After a Tx message is discarded, the flag is only raised when the CC line becomes idle.

Bit 0 **TXIS**: Transmit interrupt status

The flag indicates that the UCPD_TXDR register is empty and new data write is required (as the amount of data sent has not reached the payload size defined in the TXPAYSZ bitfield). The flag is cleared with the data write into the UCPD_TXDR register.

0: New Tx data write not required

1: New Tx data write required

51.8.6 UCPD interrupt clear register (UCPD_ICR)

Address offset: 0x018

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVTCF	Res.	Res.	Res.	Res.
											w				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2CF	TYPECEVT1CF	Res.	RXMSGENDCF	RXOVRCF	RXHRSTDETCF	RXORDDETCF	Res.	Res.	TXUNDCF	HRSTSENTCF	HRSTDISCCF	TXMSGABTCF	TXMSGSENTCF	TXMSGDISCCF	Res.
w	w		w	w	w	w			w	w	w	w	w	w	

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **FRSEVTCF**: FRS event flag (FRSEVT) clear

Setting the bit clears the FRSEVT flag in the UCPD_SR register.

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **TYPECEVT2CF**: Type-C CC2 line event flag (TYPECEVT2) clear

Setting the bit clears the TYPECEVT2 flag in the UCPD_SR register

Bit 14 **TYPECEVT1CF**: Type-C CC1 event flag (TYPECEVT1) clear

Setting the bit clears the TYPECEVT1 flag in the UCPD_SR register

Bit 13 Reserved, must be kept at reset value.

Bit 12 **RXMSGENDCF**: Rx message received flag (RXMSGEND) clear

Setting the bit clears the RXMSGEND flag in the UCPD_SR register.

Bit 11 **RXOVRCF**: Rx overflow flag (RXOVR) clear

Setting the bit clears the RXOVR flag in the UCPD_SR register.

Bit 10 **RXHRSTDETCF**: Rx Hard Reset detect flag (RXHRSTDET) clear

Setting the bit clears the RXHRSTDET flag in the UCPD_SR register.

Bit 9 **RXORDDETCF**: Rx ordered set detect flag (RXORDDET) clear
Setting the bit clears the RXORDDET flag in the UCPD_SR register.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TXUNDCF**: Tx underflow flag (TXUND) clear
Setting the bit clears the TXUND flag in the UCPD_SR register.

Bit 5 **HRSTSENTCF**: Hard reset send flag (HRSTSENT) clear
Setting the bit clears the HRSTSENT flag in the UCPD_SR register.

Bit 4 **HRSTDISCCF**: Hard reset discard flag (HRSTDISC) clear
Setting the bit clears the HRSTDISC flag in the UCPD_SR register.

Bit 3 **TXMSGABTCF**: Tx message abort flag (TXMSGABT) clear
Setting the bit clears the TXMSGABT flag in the UCPD_SR register.

Bit 2 **TXMSGSENTCF**: Tx message send flag (TXMSGSENT) clear
Setting the bit clears the TXMSGSENT flag in the UCPD_SR register.

Bit 1 **TXMSGDISCCF**: Tx message discard flag (TXMSGDISC) clear
Setting the bit clears the TXMSGDISC flag in the UCPD_SR register.

Bit 0 Reserved, must be kept at reset value.

51.8.7 UCPD Tx ordered set type register (UCPD_TX_ORDSETR)

Address offset: 0x01C

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1) and no packet transmission is in progress (TXSEND and TXHRST bits are both low).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXORDSET[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXORDSET[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **TXORDSET[19:0]**: Ordered set to transmit
The bitfield determines a full 20-bit sequence to transmit, consisting of four K-codes, each of five bits, defining the packet to transmit. The bit 0 (bit 0 of K-code1) is the first, the bit 19 (bit 4 of K-code4) the last.

51.8.8 UCPD Tx payload size register (UCPD_TX_PAYSZR)

Address offset: 0x020

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TXPAYSZ[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TXPAYSZ[9:0]**: Payload size yet to transmit

The bitfield is modified by software and by hardware. It contains the number of bytes of a payload (including header but excluding CRC) yet to transmit: each time a data byte is written into the UCPD_TXDR register, the bitfield value decrements and the TXIS bit is set, except when the bitfield value reaches zero. The enumerated values are standard payload sizes before the start of transmission.

0x2: 2 bytes - the size of Control message from the protocol layer
0x6: 6 bytes - the shortest Data message allowed from the protocol layer)
0x1E: 30 bytes - the longest non-extended Data message allowed from the protocol layer
0x106: 262 bytes - the longest possible extended message
0x3FF: 1024 bytes - the longest possible payload (for future expansion)

51.8.9 UCPD Tx data register (UCPD_TXDR)

Address offset: 0x024

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXDATA[7:0]**: Data byte to transmit

51.8.10 UCPD Rx ordered set register (UCPD_RX_ORDSETR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPKINVALID[2:0]			RXSOP3OF4	RXORDSET[2:0]		
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **RXSOPKINVALID[2:0]**:

The bitfield is for debug purposes only.
0x0: No K-code corrupted
0x1: First K-code corrupted
0x2: Second K-code corrupted
0x3: Third K-code corrupted
0x4: Fourth K-code corrupted
Others: Invalid

Bit 3 **RXSOP3OF4**:

The bit indicates the number of correct K-codes. For debug purposes only.
0: 4 correct K-codes out of 4
1: 3 correct K-codes out of 4

Bits 2:0 **RXORDSET[2:0]**: Rx ordered set code detected

0x0: SOP code detected in receiver
0x1: SOP' code detected in receiver
0x2: SOP" code detected in receiver
0x3: SOP'_Debug detected in receiver
0x4: SOP"_Debug detected in receiver
0x5: Cable Reset detected in receiver
0x6: SOP extension#1 detected in receiver
0x7: SOP extension#2 detected in receiver

51.8.11 UCPD Rx payload size register (UCPD_RX_PAYSZR)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RXPAYSZ[9:0]									
						r	r	r	r	r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **RXPAYSZ[9:0]**: Rx payload size received

This bitfield contains the number of bytes of a payload (including header but excluding CRC) received: each time a new data byte is received in the UCPD_RXDR register, the bitfield value increments and the RXMSGEND flag is set (and an interrupt generated if enabled).
0x2: 2 bytes - the size of Control message from the protocol layer
0x6: 6 bytes - the shortest Data message allowed from the protocol layer)
0x1E: 30 bytes - the longest non-extended Data message allowed from the protocol layer
0x106: 262 bytes - the longest possible extended message
0x3FF: 1024 bytes - the longest possible payload (for future expansion)
The bitfield may return a spurious value when a byte reception is ongoing (the RXMSGEND flag is low).

51.8.12 UCPD receive data register (UCPD_RXDR)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RXDATA[7:0]**: Data byte received

51.8.13 UCPD Rx ordered set extension register 1 (UCPD_RX_ORDEXTR1)

Address offset: 0x034

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXSOPX1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **RXSOPX1[19:0]**: Ordered set 1 received

The bitfield contains a full 20-bit sequence received, consisting of four K-codes, each of five bits. The bit 0 (bit 0 of K-code1) is receive first, the bit 19 (bit 4 of K-code4) last.

51.8.14 UCPD Rx ordered set extension register 2 (UCPD_RX_ORDEXTR2)

Address offset: 0x038

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXSOPX2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **RXSOPX2[19:0]**: Ordered set 2 received

The bitfield contains a full 20-bit sequence received, consisting of four K-codes, each of five bits. The bit 0 (bit 0 of K-code1) is receive first, the bit 19 (bit 4 of K-code4) last.

51.8.15 UCPD register map

Table 603. UCPD register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	UCPD_CFGR1	UCPDEN	RXDMAEN	TXDMAEN	RXORDSETEN[8:0]									PSC_UCPDCLK[2:0]			Res.	TRANSWIN[4:0]					IFRGAP[4:0]					HBITCLKDIV[5:0]						
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	UCPD_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXAFILTEN	Res.	Res.	Res.	Res.	WUPEN	FORCECLK	RXFILT2N3	RXFILTDIS
	Reset value																								0					0	0	0	0
0x008	Reserved																																
0x00C	UCPD_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2TCDIS	CC1TCDIS	Res.	RDCH	FRSTX	FRSRXEN	Res.	Res.	Res.	Res.	CCENABLE[1:0]		ANAMODE	ANASUBMODE[1:0]		PHYCCSEL	PHYRXEN	RXMODE	TXHRST	TXSEND	TXMODE[1:0]	
	Reset value											0	0		0	0	0					0	0	0	0	0	0	0	0	0	0	0	0

Table 603. UCPD register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	UCPD_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVTIE	Res.	Res.	Res.	Res.	TYPECEVT2IE	TYPECEVT1IE	Res.	RXMSGENDIE	RXOVRIE	RXHRSTDETIE	RXORDDETIE	RXNEIE	Res.	TXUNDIE	HRSTSENTIE	HRSTDISCIE	TXMSGABTIE	TXMSGSENTIE	TXMSGDISCIE	TXISIE
	Reset value												0					0	0		0	0	0	0	0		0	0	0	0	0	0	0
0x014	UCPD_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVT	TYPEC_VSTATE_CC2[1:0]		TYPEC_VSTATE_CC1[1:0]		TYPECEVT2	TYPECEVT1	RXERR	RXMSGEND	RXOVR	RXHRSTDET	RXORDDET	RXNE	Res.	TXUND	HRSTSENT	HRSTDISC	TXMSGABT	TXMSGSENT	TXMSGDISC	TXIS
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x018	UCPD_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVTCF	Res.	Res.	Res.	Res.	TYPECEVT2CF	TYPECEVT1CF	Res.	RXMSGENDCF	RXOVRCF	RXHRSTDETCF	RXORDDETCF	Res.	Res.	TXUNDCF	HRSTSENTCF	HRSTDISCCF	TXMSGABTCF	TXMSGSENTCF	TXMSGDISCCF	Res.
	Reset value												0					0	0		0	0	0	0			0	0	0	0	0	0	
0x01C	UCPD_TX_ORDSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXORDSET[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	UCPD_TX_PAYSZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXPAYSZ[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x024	UCPD_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x028	UCPD_RX_ORDSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPKINVALID[2:0]			RXSOP3OF4	RXORDSET[2:0]		
	Reset value																										0	0	0	0	0	0	0
0x02C	UCPD_RX_PAYSZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXPAYSZ[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x030	UCPD_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x034	UCPD_RX_ORDEXTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 603. UCPD register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x038	UCPD_RX_ORDEXTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C - 0x3FF	Reserved																																

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

52 Debug infrastructure

52.1 Introduction

The debug infrastructure allows software designers to debug and trace their embedded software.

The debug features can be controlled via a JTAG/Serial-wire debug access port, using industry-standard debugging tools. A trace port allows data to be captured for logging and analysis.

The trace and debug system is designed to support a variety of typical use cases:

- **Low-cost trace**

  Limited trace capability is available over the single-wire debug output. This supports code instrumentation using “printf”, tracing of data and address watchpoints, interrupt detection and program counter sampling. Single-wire trace can be maintained even when the processor is switched off or clock-stopped.
- **Breakpoint debugging**

  The processor can be debugged using equipment connected to the JTAG/SWD debug port. This allows breakpoint and watchpoint setting, code stepping, memory access and so on.
- **Tracing code execution via the trace port**

  Trace information is combined into a single trace stream and output to a trace port analyzer in real time. An ID embedded in the trace allows the analyzer to identify the source of each information packet.
- **Continuous trace capturing in a circular buffer**

  Instead of streaming it off-chip, the combined trace information can be stored on-chip in a circular buffer. The trace storage can be started and stopped by different means such as a debugger command, a software command, an external trigger signal, an internal event, and so on.
- **Draining the buffer to the trace port**

  The stored trace can be dumped off-chip to the trace port analyzer. The buffer draining can be initiated by the debugger, software, external trigger, internal event and so on.
- **Reading the buffer with the debugger**

  The debugger can read the contents of the trace buffer via the debug port. This is slower than the trace port, but allows basic trace functionality on the debugger without the cost of a trace port analyzer.
- **Analyzing stored trace in software**

  The trace buffer can be read by the processor core, or transferred into system memory by DMA. This powerful feature allows built-in test software to monitor code execution in real time, analyze and identify faults, handle exceptions autonomously, and so on.
- **Uploading stored trace**

  The stored trace can also be uploaded to a host machine using one of the MCU’s many communication interfaces (USB, USART, SPI, I2C, Ethernet, CAN and so on). This is especially useful if the trace port is not accessible, for example remote monitoring and failure analysis of a deployed product.

52.2 Debug infrastructure features

A comprehensive set of trace and debug features is provided to support software development and system integration:

- Breakpoint debugging
- Code execution tracing
- Software instrumentation
- Cross-triggering
- JTAG debug port
- Serial-wire debug port
- Trigger input and output
- Serial-wire trace port
- Trace port
- Arm® CoreSight™ debug and trace components.

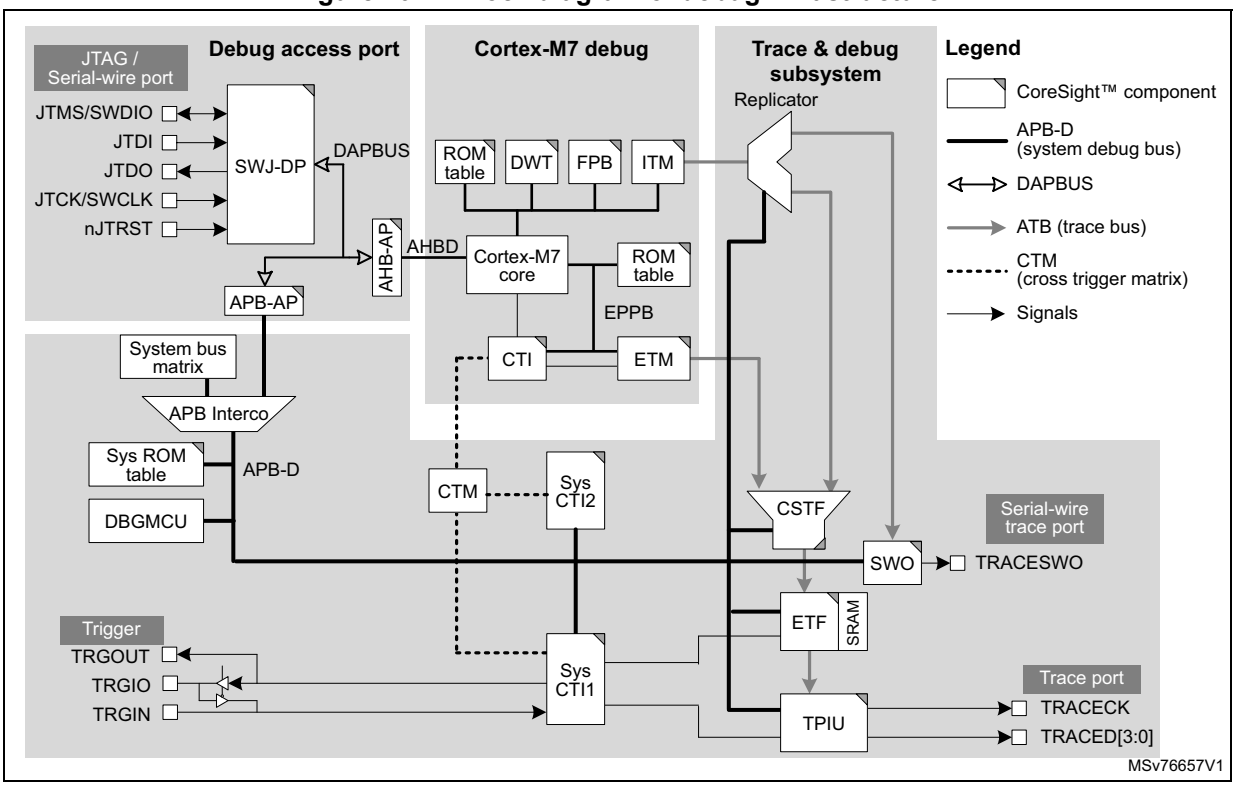
The CoreSight components are described at high level in this document. Detailed information is available in the Arm® documents referenced in *Section 52.11*.

52.3 Debug infrastructure functional description

52.3.1 Debug infrastructure block diagram

The block diagram shows the logical partitioning of the debug infrastructure.

Figure 1042. Block diagram of debug infrastructure



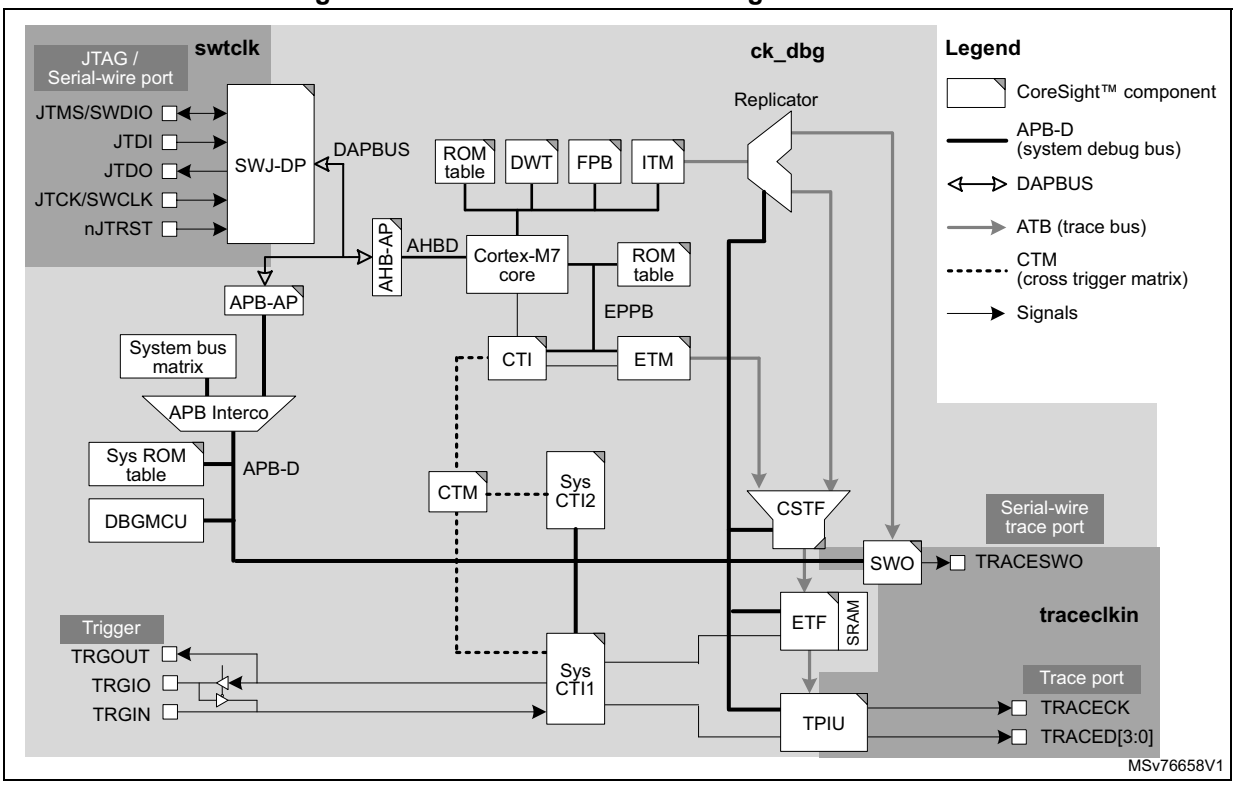
52.3.2 Debug infrastructure powering, clocking and reset

Power domains

The debug components are all located in the core power domain. This means that a debug session is disconnected if the core is powered down, for example in standby mode. This can be prevented by enabling low-power emulation mode via the DBGMCU_CR.DBG_STANDBY register bit.

Clock domains

Figure 1043. Clock domains of debug infrastructure



The debugger supplies the clock for the debug port, swtclk, via the debug interface pin, JTCK/SWCLK. This clock is used to register the serial input data in both serial wire and JTAG mode, as well as to operate the state machines and internal logic of the debug port. It must therefore continue to toggle for several cycles after the end of an access, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the ck_dbg domain, which covers the rest of the SWJ-DP and the access ports.

ck_dbg clocks the trace components on the debug bus (APB-D): System ROM table, CoreSight trace funnel, ETF, system CTIs and TPIU. It is a gated version of the core domain system clock (rcc_hclk3).

traceclkin is the trace port clock.

The debug clocks (ck_dbg and traceclkin) can be enabled and disabled by register bits in the DBGMCU (DBGMCU_CR.DBGCKEN and DBGMCU_CR.TRACECLKEN) or by the debugger using the CDBGPWRUPREQ bit in the debug port CTRLSTAT register. These

clocks must be enabled before the debugger or software can access any of the debug features on the APB-D. They are disabled at power-up and can be disabled when the debugger is disconnected, to avoid wasting energy.

The debug and trace components included in the processor (ETM ITM, DWG, FPB and so on) are clocked with the corresponding core clock (rcc_c_ck).

Debug with low-power modes

The device includes power-saving features allowing individual power domains to be switched off or stopped when not required. If a power domain is switched off or not clocked, all debug components in that domain are inaccessible to the debugger. To avoid this, power-saving mode emulation is implemented. If the emulation is enabled for a domain, the domain still enters power-saving mode, but its clock and power are maintained. In other words, the domain behaves as if it is in power-saving mode, while the debugger does not lose the connection.

The emulation mode is programmed in the MCU Debug (DBGMCU) unit. For more information, refer to *Section 52.8*

Reset of debug infrastructure

The debug components, including for the debug port and access ports, are only reset by a power on reset. Thus, during a system reset the debug session remains alive. The nJTRST pin resets only the JTAG TAP state machine.

52.3.3 Security

The trace and debug components allow a high degree of access to the processor and system during product development. In order to protect user code and ensure that the debug features cannot be used to alter or compromise the normal operation of the finished product, these features can be disabled or limited in scope.

Debugger access is disabled while the processor is booting from system flash memory.

The following authentication signals are used by the system to determine which debug features are enabled or disabled:

- **dbgen**: global enable for all debug features
  0: All debug features are disabled.
  1: Debug features are enabled.
- **niden**: enables trace and performance monitoring (non-invasive debug).
  0: Trace generation is disabled.
  1: Trace generation is enabled.

For detailed information on the behavior of each component according to the state of the authentication signals, refer to the relevant component chapter or to the relevant Arm® technical documentation.

The state of the signals is set according to the debug state as shown in *Table 604*.

Table 604. Authentication signal states

Debug state	Authentication signal state	Description
OPEN	dbgen = 1 niden = 1	Debug and trace is enabled whatever the state of the processor.
CLOSED	dbgen = 0 niden = 0	Debug and trace is disabled.

The debug state depends on the product life-cycle (see *Section 10.3.6: SBS debug management*).

52.4 Serial-wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a CoreSight component that implements an external access port for connecting debugging equipment.

The port can be configured as:

- A 5-pin standard JTAG debug port (JTAG-DP)
- A 2-pin (clock + data) “serial-wire” debug port (SW-DP)

The two modes are mutually exclusive, since they share the same IO pins.

By default, the JTAG-DP is selected on system or power-on reset. The five IOs are configured by hardware in debug alternative function mode. The SWJ-DP incorporates pull-up resistors on the JTDI, JTMS/SWDIO, and nJTRST lines, as well as a pull-down resistor on the JTCK/SWCLK line.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO: ...,(50 or more ones),...,0,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1... (50 or more ‘1’s)...

JTCK/SWCLK must be cycled for each data bit.

In SW-DP mode, the unused JTAG lines JTDI, JTDO and nJTRST can be used for other functions.

All SWJ port IOs can be reconfigured to other functions by software, in which case debugging is no longer possible.

52.4.1 Serial wire debug port

The serial wire debug protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bi-directional serial data

Serial data is transferred LSB first, synchronously with the clock. A transfer comprises three phases:

1. Packet request (8 bits) transmitted by the host
2. Acknowledge response (3 bits) transmitted by the target
3. Data transfer (33 bits) transmitted by the host (in the case of a write) or target (in the case of a read)

The data transfer only occurs if the acknowledge response is OK.

If the direction of the data is reversed between each phase, a single clock cycle turn-around time is inserted.

Table 605. Packet request

Field bits	Name	Description
0	Start	Must be “1”
1	APnDP	0: DP register access - see *Table 609* for a list of DP registers 1: AP register access - see *Section 52.5*
2	RnW	0: Write request 1: Read request
4:3	A(3:2)	Address field of the DP or AP register (refer to *Table 609* and *Table 611*)
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by host. Must be read as “1” by target.

Table 606. ACK response

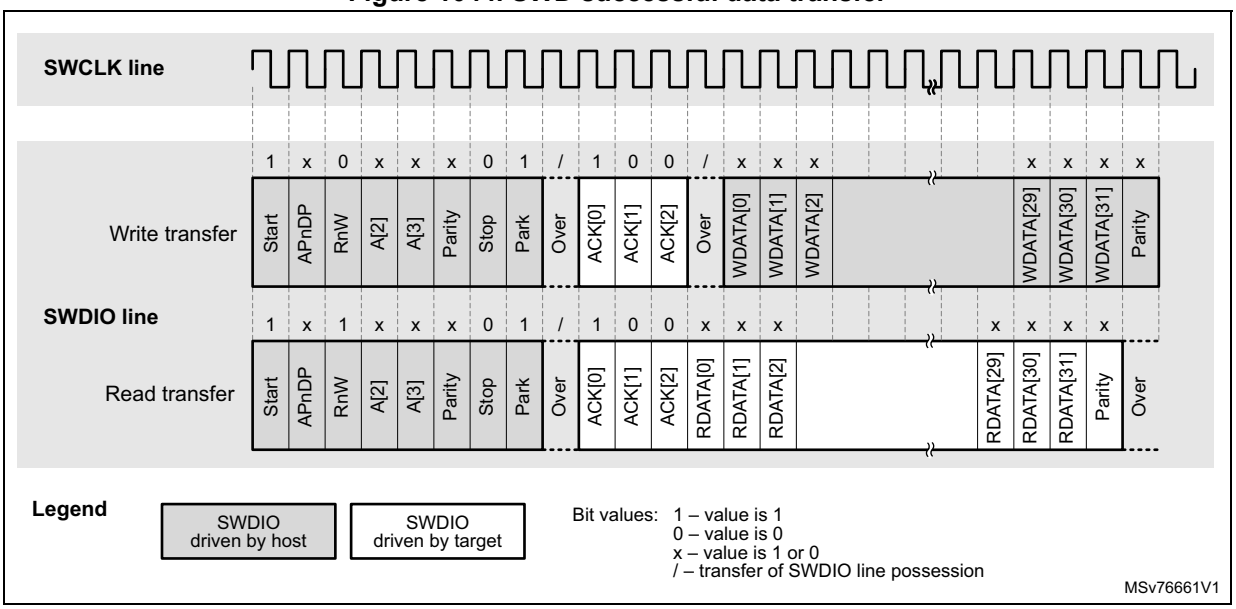
Field bits	Name	Description
2:0	ACK	000b: Fault 010b: Wait 100b: OK

Table 607. Data transfer

Bit field	Name	Description
31:0	WDATA or RDATA	Write or Read data
32	Parity	Single bit parity of 32 data bits

Figure 1044 shows successful write and read transfers.

Figure 1044. SWD successful data transfer



For any FAULT or WAIT ACK response from the target, the data transfer phase is canceled, unless overrun detection is enabled, in which case the data is ignored by the target (in the case of a write), or not driven (in the case of a read).

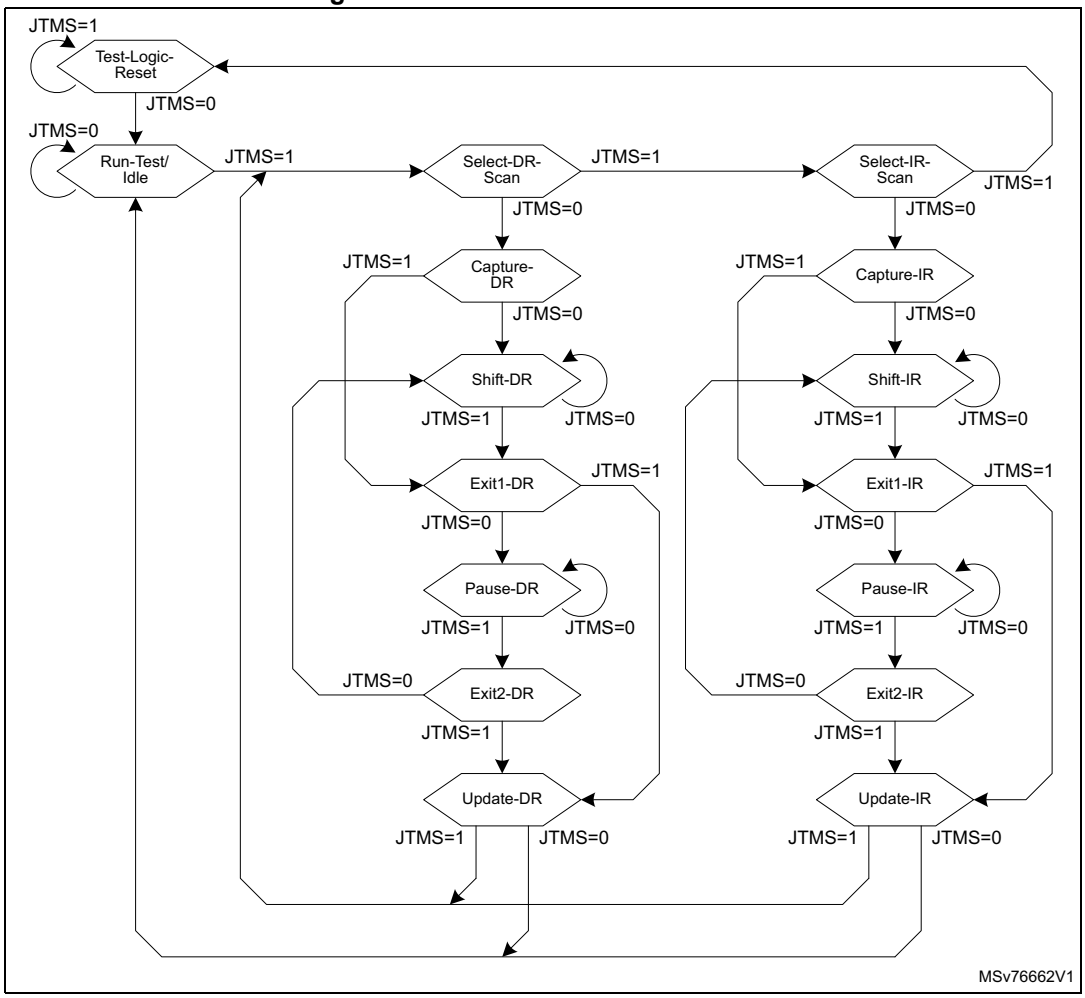
A line reset must be generated by the host when it is first connected, or following a protocol error. The line reset consists of 50 or more SWCLK cycles with SWDIO high, followed by two SWCLK cycles with SWDIO low.

For more details on the Serial Wire debug protocol, refer to the Arm® Debug Interface Architecture Specification *[1]*.

Note: The SWJ-DP implements SWD protocol version 2.

52.4.2 JTAG debug port

Figure 1045. JTAG TAP state machine



The JTAG-DP implements a TAP state machine (TAPSM) based on IEEE 1149.1-1990. The state machine is shown in *Figure 1045*. It controls two scan chains, one associated with an instruction register (IR) and one with a number of data registers (DR).

When the TAPSM goes through the Capture-IR state, 0b0001 is transferred into the instruction register (IR) scan chain. The IR scan chain is connected between JTDI and JTDO.

While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each rising edge of JTCK. This means that on the first tick:

- The LSB of the IR scan chain is output on JTDO.
- Bit [n] of the IR scan chain is transferred to bit [n-1].
- The value on JTDI is transferred to the MSB of the IR scan chain.

When the TAPSM goes through the Update-IR state, the value scanned in the IR scan chain is transferred to the instruction register.

When the TAPSM goes through the Capture-DR state, a value is transferred from one of the data registers onto one of the DR scan chains, connected between JTDI and JTDO.

The value held in the instruction register determines which data register, and associated DR scan chain, is selected.

This data is then shifted while the TAPSM is in the Shift-DR state, in the same way as the IR shift in the Shift-IR state.

When the TAPSM goes through the Update-DR state, the value scanned in the DR scan chain is transferred to the selected data register.

When the TAPSM is in the Run-Test/Idle state, no special actions occur. The IDCODE instruction is loaded in IR.

When active, the nJTRST signal resets the state machine asynchronously to the Test-Logic-Reset state.

The data registers corresponding to the 4-bit IR instructions are listed in *Table 608*.

Table 608. JTAG-DP data registers

Instruction register	Data register	Scan chain length	Description
0000 to 0111	(BYPASS)	1	Not implemented: BYPASS selected
1000	ABORT	35	**Abort register** – Bits 34:1 = reserved – Bit 0 = APABORT: write 1 to generate an AP abort
1001	(BYPASS)	1	Reserved: BYPASS selected
1010	DPACC	35	**Debug port access register** Initiates the debug port and allows access to a debug port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request Bits 2:1 = A[3:2] = 2-bit address of a debug port register. Bit 0 = RnW = Read request (1) or write request (0). – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010b = OK/fault 001b = Wait Other = reserved

Table 608. JTAG-DP data registers (continued)

Instruction register	Data register	Scan chain length	Description
1011	APACC	35	**Access port access register** Initiates an access port and allows access to an access port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request Bits 2:1 = A[3:2] = 2-bit sub-address of an access port register. Bit 0 = RnW= Read request (1) or write request (0). – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010b = OK/Fault 001b = Wait Other = reserved
1100	(BYPASS)	1	Reserved: BYPASS selected
1101	(BYPASS)	1	Reserved: BYPASS selected
1110	IDCODE	32	**ID Code** 0x6BA00477: Arm® JTAG debug port ID code
1111	BYPASS	1	**Bypass** A single JTCK cycle delay is inserted between JTDI and JTDO

The DR registers are described in more detail in the Arm® Debug Interface Architecture Specification *[1]*.

52.4.3 Debug port registers

The SW-DP and JTAG-DP both access the debug port (DP) registers. These are listed in *Table 609*.

The debugger can access the DP registers as follows:

1. Program the SELECT register DPBANKSEL field in the DP to select the register bank to be accessed (see *Table 609*).
2. Program the A[3:2] field in the DPACC register, if using JTAG, with the register address within the bank. Program the R/W bit to select a read or a write. In the case of a write, program the DATA field with the write data. If using SWD, the A[3:2] and R/W fields are part of the Packet Request word sent to the SW-DP with the APnDP bit reset (see *Table 605*). The write data is sent in the data phase.

Table 609. Debug port registers

Address	A[3:2] field value	R/W	Description
0x0	00	R	DP_DPIDR register(2). Contains the IDCODE for the debug port.
		W	DP_ABORT register(1). Aborts the current AP transaction. This register is also used to clear the error flags in the DP_CTRL/STAT register.
0x4	01	R/W	If DPBANKSEL[3:0] = 0x0 (DP_SELECT register): CTRLSTAT register. Controls the DP and provides status information.
			If DPBANKSEL[3:0] = 0x1 (DP_SELECT register): DP_DLCR register(2). Controls the operating mode of the SWD Data Link.
			If DPBANKSEL[3:0] = 0x2 (DP_SELECT register): DP_TARGETID register. Provides target identification information.
			If DPBANKSEL[3:0] = 0x3 (DP_SELECT register): DLPIDR register(2). Provides the SWD protocol version.
0x8	10	R	RESEND register(2). Returns the value that was returned by the last AP read or DP_RDBUFF read, used in the event of a corrupted read transfer.
		W	DP_SELECT register. Selects the access port, access port register bank, and DP register at address 0x4.
0xC	11	R	DP_RDBUFF register Via JTAG-DP, enables the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation). Via SW-DP, contains the result of the preceding AP read access, allowing a new AP access to be avoided.
		W	DP_TARGETSEL register(2). On a write to DP_TARGETSEL immediately following a line reset sequence, the target is selected if the following conditions are both met: – Bits [31:28] match bits [31:28] in the DP_DLPIDR register. – Bits [27:0] match bits [27:0] in the DP_TARGETID register. Writing any other value deselects the target. Debug tools must write 0xFFFFFFFF to deselect all targets. This is an invalid DP_TARGETID value. All other invalid DP_TARGETID values are reserved.

1. Access to the AP ABORT register from the JTAG-DP is done using the ABORT instruction.
2. Only accessible via SW-DP. Register is “reserved” via JTAG-DP.

Debug port identification register (DP_DPIDR)

Address offset: 0x0

Reset value: 0x6BA0 2477

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN
r	r	r	r	r	r	r	r	r	r	r	r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION[3:0]				DESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **REVISION[3:0]**: Revision code
0x6

Bits 27:20 **PARTNO[7:0]**: Debug port part number
0xBA

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **MIN**: Minimal debug port (MINDP) implementation
0: MINDP not implemented (transaction counter and pushed operations are supported)

Bits 15:12 **VERSION[3:0]**: DP architecture version
0x2: DPv2

Bits 11:1 **DESIGNER[10:0]**: JEDEC designer identity code
0x23B: Arm®

Bit 0 Reserved, must be kept at reset value.

Debug port abort register (DP_ABORT)

Address offset: 0x0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUN ERRCL R	WDER RCLR	STKER RCLR	STKCM PCLR	DAPAB ORT
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ORUNERRCLR**: Overrun error clear bit
0: No effect
1: Clear CTRLSTAT register's STICKYORUN bit

Bit 3 **WDERRCLR**: Write data error clear bit
0: No effect
1: Clear CTRLSTAT register's WDATAERR bit

Bit 2 **STKERRCLR**: Sticky error clear bit
0: No effect
1: Clear CTRLSTAT register's STICKYERR bit

Bit 1 **STKCMPCLR**: Sticky compare clear bit
0: No effect
1: Clear CTRLSTAT register's STICKYCMP bit

Bit 0 **DAPABORT**: Abort current AP transaction
The transaction is aborted if an excessive number of WAIT responses are returned, indicating that the transaction has stalled.
0: No effect
1: Abort transaction

Debug port control/status register (DP_CTRLSTAT)

Address offset: 0x4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSYSPWRUPACK	CSYSPWRUPREQ	CDBGPWRUPACK	CDBGPWRUPREQ	CDBGRSTACK	CDBGRSTREQ	Res.	Res.	TRNCNT[11:4]							
r	rw	r	rw	r	rw			rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRNCNT[3:0]				MASKLANE[3:0]				WDATAERR	READOK	STICKYERR	STICKYCMP	TRNMODE[1:0]		STICKYORUN	ORUNDETECT
rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	w	w	w	w	w

Bit 31 **CSYSPWRUPACK**: System domain power-up status bit - not used in this device

Bit 30 **CSYSPWRUPREQ**: System domain power-up control bit - not used in this device

Bit 29 **CDBGPWRUPACK**: Debug domain power-up status bit
This bit is read-only. It returns the status of the debug clock.
0: Debug clock disabled
1: Ddebug clock enabled

Bit 28 **CDBGPWRUPREQ:** Debug clock control bit
This bit controls the debug clock request.
0: Clock not requested
1: Clock requested

Bit 27 **CDBGRSTACK**: Debug domain reset status bit - not used in this device

Bit 26 **CDBGRSTREQ**: Debug domain reset control bit - not used in this device

Bits 25:24 Reserved, must be kept at reset value.

Bits 23:12 **TRNCNT[11:0]**: Transaction counter
To program a sequence of transactions to incremental addresses via an AP, TRNCNT bits are loaded with the number of transactions to perform. It is decremented on successful completion of each transaction.

Bits 11:8 **MASKLANE[3:0]**: Pushed-compare and pushed-verify masking bits

The field indicates the bytes to be masked in pushed-compare and pushed-verify operations (DP_CTRLSTAT register's field TRNMODE = 1 or 2). In the pushed operations, the word supplied in an AP write transaction is compared with the current value at the target AP address.

0b1XXX: Include byte lane 3 in comparisons

0bX1XX: Include byte lane 2 in comparisons

0bXX1X: Include byte lane 1 in comparisons

0bXXX1: Include byte lane 0 in comparisons

Bit 7 **WDATAERR**: Write data error in SW-DP

The bit indicates;

- A parity or a framing error on the data phase of a write operation, or
- A write operation that had been accepted by the DP has then been discarded without being submitted to the AP

This bit is read-only. It is reset by writing 1 to the WDERRCLR bit of the DP_ABORT register.

0: No error

1: Error has occurred

This bit is reserved in JTAG-DP.

Bit 6 **READOK**: AP read response in SW-DP

This bit indicates the response to the last AP read access. It is read-only.

0: Read not OK

1: Read OK

This bit is Reserved in JTAG-DP.

Bit 5 **STICKYERR**: Transaction error (read-only in SW-DP, R/W in JTAG-DP)

This bit indicates that an error occurred during an AP transaction.

0: No error

1: Error has occurred

In the SW-DP, this bit is reset by writing 1 to the STKERRCLR bit of the DP_ABORT register.
In the JTAG-DP, this bit is reset by programming it to 1.

Bit 4 **STICKYCMP**: Compare match (read-only in SW-DP, R/W in JTAG-DP)

This bit indicates that a match occurred in a pushed operation.

0: Match if TRNMODE = 0x1; no match if TRNMODE = 0x2

1: No match if TRNMODE = 0x1; match if TRNMODE = 0x2

In the SW-DP, this bit is reset by writing 1 to the STKCMPCLR bit in the DP_ABORT register.
In the JTAG-DP, this bit is reset by programming it to 1.

Bits 3:2 **TRNMODE[1:0]**: Transfer mode for AP write operations

For read operations, this field must be set to 0x0.

0x0: Normal operation - AP transactions are passed directly to the AP.

0x1: Pushed-verify operation. The DP stores the write data and performs a read transaction at the target AP address. The result of the read operation is compared with the stored data. If they do not match, the STICKYCMP bit is set.

0x2: Pushed-compare operation. The DP stores the write data and performs a read transaction at the target AP address. The result of the read is compared with the stored data. If they match, the STICKYCMP bit is set.

0x3: Reserved

In pushed operations, only the data bytes indicated by the MASKLANE field are included in the comparison.

Bit 1 **STICKYORUN:** Overrun (read-only in SW-DP, R/W in JTAG-DP)

This bit indicates that an overrun occurred (a new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set.

0: No overrun

1: Overrun occurred

In the SW-DP, this bit is reset by writing 1 to the ABORT register's ORUNERRCLR bit. In the JTAG-DP, this bit is reset by writing a 1 to it.

Bit 0 **ORUNDETECT:** Overrun detection mode enable

0: Overrun detection disabled

1: Overrun detection enabled. In the event of an overrun, the STICKYORUN bit is set and subsequent transactions are blocked until the STICKYORUN bit is cleared.

Debug port data link control register (DP_DLCR)

Address offset: 0x4

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw	rw								

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **TURNROUND[1:0]**: Tristate period for SWDIO

0x0: 1 data bit period

0x1: 2 data bit periods

0x2: 3 data bit periods

0x3: 4 data bit periods

Bits 7:0 Reserved, must be kept at reset value.

Debug port target identification register (DP_TARGETID)

Address offset: 0x4

Reset value: 0x147B 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TREVISION[3:0]				TPARTNO[15:4]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPARTNO[3:0]				TDESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **TREVISION[3:0]**: Device revision number
0x1

Bits 27:12 **TPARTNO[15:0]**: Target part number
0x47B0: STM32H7Pxxx

Bits 11:1 **TDESIGNER[10:0]**: Target designer JEDEC code
0x020: STMicroelectronics

Bit 0 Reserved, must be kept at reset value.

Debug port data link protocol identification register (DP_DLPIDR)

Address offset: 0x4

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TINSTANCE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]			
												r	r	r	r

Bits 31:28 **TINSTANCE[3:0]**: Target instance number
These bits define the instance number for this device in a multi-drop system.
0x0

Bits 27:4 Reserved, must be kept at reset value.

Bits 3:0 **PROTSVN[3:0]**: Serial Wire Debug protocol version
0x1: Version 2

Debug port resend register (DP_RESEND)

Address offset: 0x8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESEND[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESEND[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RESEND[31:0]**: Last AP read or DP RDBUFF read value

These bits contain the value that was returned by the last AP read or DP RDBUFF read. Used in the event of a corrupted read transfer.

Debug port access port select register (DP_SELECT)

Address offset: 0x8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APSEL[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w	w	w	w	w	w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
								w	w	w	w	w	w	w	w

Bits 31:24 **APSEL[7:0]**: Access port select bits

These bits select the access port for the next transaction.

0x00: AP0 - System debug access port (APB-AP)

0x01: AP1 - Cortex-M7 debug access port (AHB-AP)

0x02 to 0x1F: Reserved

Bits 23:8 Reserved, must be kept at reset value.

Bits 7:4 **APBANKSEL[3:0]**: AP register bank select bits

These bits select the 4-word register bank on the active AP for the next transaction.

Bits 3:0 **DPBANKSEL[3:0]**: DP register bank select bits

These bits select the register at address 0x4 of the debug port.

0x0: CTRLSTAT register

0x1: DLCR register

0x2: TARGETID register

0x3: DLPIDR register

0x4 to 0xF: Reserved

Debug port read buffer register (DP_RDBUFF)

Address offset: 0xC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDBUFF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDBUFF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDBUFF[31:0]**: Last AP read value

The field contains the value returned by the last AP read access. There are two ways to retrieve the value returned by an AP read access:

- Perform a second read access to the same address, which initiates a new transaction on the corresponding bus, or
- Read the value returned by the last AP read access from the DP_RDBUFF register, in which case no new AP transaction occurs

Debug port target identification register (DP_TARGETSEL)

Address offset: 0xC

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TINSTANCE[3:0]				TPARTNO[15:4]											
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPARTNO[3:0]				TDESIGNER[10:0]											Res.
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	

Bits 31:28 **TINSTANCE[3:0]**: Target instance number

The field defines the instance number for the target device in a multi-drop system. It must be programmed with the same value as TINSTANCE field of DP_DLPIDR register, in order to select this device.

Bits 27:12 **TPARTNO[15:0]**: Target part number

The field defines the part number for the target device. It must be programmed with the same value as TPARTNO field of DP_TARGETID register, in order to select this device.

Bits 11:1 **TDESIGNER[10:0]**: Target designer JEDEC code

The field defines the JEDEC code for the target device. It must be programmed with the same value as TDESIGNER field of DP_TARGETID register, in order to select this device.

Bit 0 Reserved, must be kept at reset value.

52.4.4 Debug port register map

These registers are not on the CPU memory bus, they are only accessed through SW-DP and JTAG-DP debug interface.

The debug port address offset is 4 bits wide, where the 2 most significant bits are defined in the JTAG-DP register DPACC or SW-DP packet request A[3:2] field. The 2 least significant bits are 00.

Table 610. Debug port register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	DP_DPIDR	REVISION[3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN	VERSION[3:]				DESIGNER[10:0]											Res.
	Reset value	0	1	1	0	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	1	0	1	1	1
0x0	DP_ABORT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	STKCMPCLR	DAPABORT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4(1)	DP_CTRLSTAT	CSYSPWRUPACK	CSYSPWRUPREQ	CDBGPWRUPACK	CDBGPWRUPREQ	CDBGRSTACK	CDBGRSTREQ	Res.	Res.	TRNCNT[11:0]												MASKLANE[3:0]				WDATAERR	READOK	STICKYERR	STICKYCMP	TRNMODE[1:0]		STICKYORUN	ORUNDETECT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4(2)	DP_DLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0x4(3)	DP_TARGETID	TREVISION[3:0]				TPARTNO[15:4]												TPARTNO[3:0]				TDESIGNER[10:0]											Res.
	Reset value	0	0	0	1	0	1	0	0	0	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
0x4(4)	DP_DLPIDR	TINSTANCE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x8	DP_RESEND	RESEND[31:0]]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

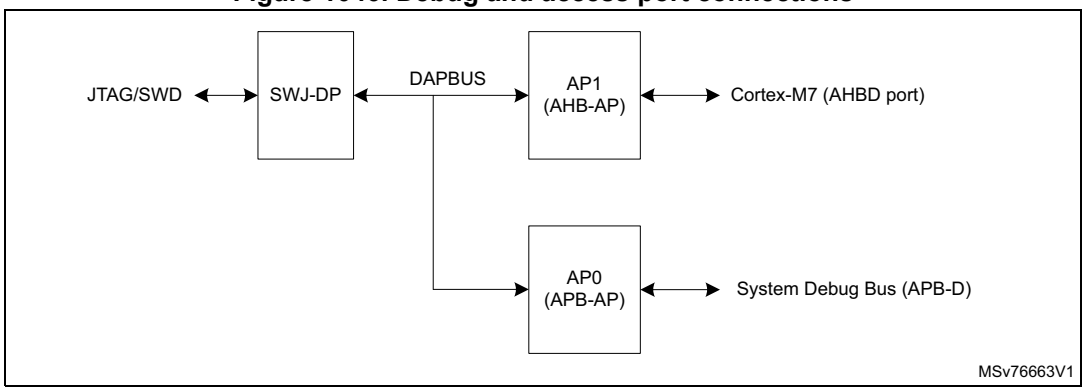
Table 610. Debug port register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x8	DP_SELECT	APSEL[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0xC	DP_RDBUFF	RDBUFF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC	DP_TARGETSEL	TINSTANCE[3:0]				TPARTNO[15:4]												TPARTNO[3:0]				TDESIGNER[10:0]											Res.
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

1. DPBANKSEL[3:0] = 0x0.
2. DPBANKSEL[3:0] = 0x1.
3. DPBANKSEL[3:0] = 0x2.
4. DPBANKSEL[3:0] = 0x3.

52.5 Access ports

Figure 1046. Debug and access port connections



Access ports (AP) attached to the DP:

1. AP1: Cortex-M7 access port (AHB-AP). Allows access to the debug and trace features integrated in the Cortex-M7 processor core via an AHB-Lite bus connected to the AHBD port of the processor.
2. AP0: System access port (APB-AP). Allows access to the debug and trace features on the system APB debug bus, that is, all components not included in the processor core.

All access ports are of MEM-AP type, that is, the debug and trace component registers are mapped in the address space of the associated debug bus. The AP is seen by the debugger as a set of 32-bit registers organized in banks of four registers each. Some of these registers are used to configure or monitor the AP itself, while others are used to perform a transfer on the bus. The AP registers are listed in *Table 611*.

The address of the AP registers is composed of:

- Bits [7:4]: content of the DP_SELECT register's APBANKSEL field
- Bits [3:2]: content of the A(3:2) field of the APACC data register in the JTAG-DP (see *Table 608*) or of the SW-DP Packet Request (see *Table 605*), depending on the debug interface used
- Bits [1:0]: Always set to 0

The content of the SELECT register APSEL field in the DP defines which MEM-AP is being accessed.

The debugger can access the AP registers as follows:

1. Program the DP_SELECT register's APSEL field to choose one of the APs, and the APBANKSEL field to select the register bank to be accessed.
2. Program the A(3:2) field in the APACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or a write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the Packet Request word sent to the SW-DP with the APnDP bit set (see *Table 605*). The write data is sent in the data phase.

The debugger can access the memory-mapped debug component registers through the MEM-AP registers (using the AP register access procedure described above) as follows:

1. Program the transaction target address in the TAR register.
2. Program the CSW register, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from the DRW register to initiate a bus transaction at the address held in the TAR register. Alternatively, a read or write to banked data register BDn triggers an access to address TAR[31:4] + n (this allows an access to the next four consecutive addresses without changing the address in the TAR register).

For more detailed information on the MEM-AP, refer to the Arm® Debug Interface Architecture Specification *[1]*.

52.5.1 MEM-AP registers

Table 611. MEM-AP registers

Address	APBANKSEL	A(3:2)	Name	Description
0x00	0x0	0	AP_CSW	Control/status word register
0x04	0x0	1	AP_TAR	Transfer address register Target address for the bus transaction.
0x08	-	-	-	Reserved
0x0C	0x0	3	AP_DRW	Data read/write register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:0]
0x10	0x1	0	AP_BD0	Banked data 0 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] << 4 + 0x0
0x14	0x1	1	AP_BD1	Banked Data 1 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] << 4 + 0x4
0x18	0x1	2	AP_BD2	Banked data 2 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] << 4 + 0x8
0x1C	0x1	3	AP_BD3	Banked data 3 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] << 4 + 0xC
0x20-0xEC	-	-	-	Reserved
0xF0	-	-	-	Reserved
0xF4	-	-	-	Reserved
0xF8	0xF	2	AP_BASE	Debug base address register (RO) Base address of the ROM table
0xFC	0xF	3	AP_IDR	Identification register (RO)

Access port control/status word register (AP_CSW)

Address offset: 0x0

Reset value: 0x0000 0002 (APB-AP), 0x4000 0002 (AHB-AP)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SPROT	Res.	PROT[4:0]					SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw	rw	rw	rw	r							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DEVICEEN	ADDRINC[1:0]		Res.	SIZE[2:0]		
				rw	rw	rw	rw	r	r	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SPROT**: Secure transfer request bit

In the APB-AP, this field is reserved. In the AHB-APs, this field sets the protection attribute HPROT[6] of the bus transfer.

0: Reserved.

1: Non-secure transfer.

Bit 29 Reserved, must be kept at reset value.

Bits 28:24 **PROT[4:0]**: Bus transfer protection bits

In the APB-AP, this field is reserved. In the AHB-APs, this field sets the protection attributes HPROT[4:0] of the bus transfer.

0bXXXX0: Instruction fetch

0bXXXX1: Data access

0bXXX0X: User mode

0bXXX1X: Privileged mode

0bXX0XX: Non-bufferable

0bXX1XX: Bufferable

0bX0XXX: Non-cacheable

0bX1XXX: Cacheable

0b0XXXX: Non-exclusive

0b1XXXX: Exclusive

Bit 23 **SPISTATUS**: Status of SPIDEN authentication signal

This bit determines whether the debugger can access secure memory. This field is reserved in the APB-AP.

0: Secure AHB transfers are not supported

1: Not used

Bits 22:12 Reserved, must be kept at reset value.

Bits 11:8 **MODE[3:0]**: Barrier support enabled bit

These bits define if the memory barrier operation is supported.

0x0: Not supported

Bit 7 **TRINPROG**: Transfer in progress

This bit indicates that an AP bus transfer is in progress.

0: No transfer in progress.

1: Bus transfer in progress.

Bit 6 **DEVICEEN**: Device Enable bit
This bit defines whether the AP can be accessed or not.
1: AP access enabled.

Bits 5:4 **ADDRINC[1:0]**: Auto-increment mode bits
These bits define whether the TAR address is automatically incremented after a transaction.
0x0: No auto-increment
0x1: Address is incremented by the size in bytes of the transaction (SIZE field).
0x2: Packed transfers enabled (Only in AHB-APs - reserved in APB-AP). A 32-bit AP access generates a 1 x 32-bit, 2 x 16-bit or 4 x 8-bit bus transaction corresponding to the programmed transaction size. The data is packed or unpacked accordingly.
0x3: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: Size of next memory access transaction (only for AHB-APs)
0x0: Byte (8-bit)
0x1: Half-word (16-bit)
0x2: Word (32-bit)
0x3-0x7: Reserved
For APB-AP, this field is read-only and fixed at 0x2 (32-bit).

AP transfer address register (AP_TAR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRESS[31:0]**: Address of the current transfer (bits 31:0)

AP data read/write register (AP_DRW)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Read/write data for current transfer
In write mode, this is the write data value. In read mode, this is the read data value.

AP banked data register 0 (AP_BD0)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Read/write data for current transfer
The transaction address is TAR[31:4] << 4 + 0x0.

AP banked data register 1 (AP_BD1)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Read/write data for current transfer
The transaction address is TAR[31:4] << 4 + 0x4.

AP banked data register 2 (AP_BD2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Read/write data for current transfer
The transaction address is TAR[31:4] << 4 + 0x8.

AP banked data register 3 (AP_BD3)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Read/write data for current transfer
The transaction address is TAR[31:4] << 4 + 0xC.

Access port base address register (AP_BASE)

Address offset: 0xF8

Reset value: 0xE00F E003 (AP1), 0xE00E 0003 (AP0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASEADDR[19:4]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASEADDR[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
r	r	r	r											r	r

Bits 31:12 **BASEADDR[19:0]**: Base address (bits 31 to 12) of the AP ROM table
The 12 LSBs are zero since the ROM table must be aligned on a 4 Kbyte boundary.
AP1 (Cortex-M7 AHB-AP): 0xE00FE
AP0 (System APB-AP): 0xE00E0

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **FORMAT**: Base address register format
1: Arm® debug interface v5.

Bit 0 **ENTRYPRESENT**: Debug component present status bit
This bit indicates that debug components are present on the access port bus:
1: Debug components are present

Access port identification register (AP_IDR)

Address offset: 0xFC

Reset value: 0x8477 0001 (AP1), 0x5477 0002 (AP0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							MEMAP
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:28 **REVISION[3:0]**: Arm core revision
0x5: r1p0 (AP0)
0x8: r0p9 (AP1)

Bits 27:24 **JEDECBANK[3:0]**: JEDEC bank
0x4: Arm®

Bits 23:17 **JEDECCODE[6:0]**: JEDEC code
0x3B: Arm®

Bit 16 **MEMAP**: Memory access port
1: Standard register map

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **IDENTITY[7:0]**: AP type identification
0x01: AHB-AP (AP1)
0x02: APB-AP (AP0)

52.5.2 Access port register map

These registers are not on the CPU memory bus, they are only accessed through SW-DP and JTAG-DP debug interfaces.

The access port address is 8-bit wide, defined by DP_SELECTR.APBANKSEL[3:0] field and by JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

Table 612. Access port register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	AP_CSW	Res.	SPROT	Res.	PROT[4:0]					SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DEVICEEN	ADDRINC[1:0]		Res.	SIZE[2:0]		
	Reset value		X		0	0	0	0	0	0												0	0	0	0	0	0	0	0		0	1	0
0x04	AP_TAR	ADDRESS[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	Reserved																																
0x0C	AP_DRW	DATA[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	AP_BD0	DATA[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	AP_BD1	DATA[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	AP_BD2	DATA[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	AP_BD3	DATA[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20 to 0xF4	Reserved																																

Table 612. Access port register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xF8	AP_BASE	BASEADDR[19:4]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
	Reset value	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	X	0	0	0	0											1	1
0xFC	AP_IDR	REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							MEMAP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
	Reset value	X	X	X	X	0	1	0	0	0	1	1	1	0	1	1	1									0	0	0	0	0	0	X	X

52.6 Trace and debug subsystem functional description

The trace and debug subsystem features the following CoreSight components:

- System ROM tables
- System cross-trigger interface (CTI)
- Cross-trigger matrix (CTM)
- Trace port interface unit (TPIU)
- Trace bus funnel (CSTF)
- Embedded trace FIFO (ETF)
- Serial wire output (SWO)

These components are accessible by the debugger via the system APB-AP and its associated APB-D debug bus. They are also accessible by the Cortex-M7 processor.

The MCU debug unit (DBGMCU) is also accessed via the APB-D. This component contains registers for configuring the device behavior in Debug mode.

A trace bus replicator branches the trace bus from the CPU's ITM CoreSight component to ETF and SWO, through trace bus funnels.

52.6.1 System ROM table

There is a ROM table on the APB-D bus. The ROM table is a CoreSight component that contains the base addresses of all the CoreSight components on the APB-D bus. These tables allow a debugger to discover the topology of the CoreSight components automatically.

The table points to the following CoreSight components: SWO, DBGMCU, System-CTI1, System-CTI2, Trace Funnel, TPIU, Replicator and ETF. The table occupies a 4-Kbyte, 32-bit wide chunk of the APB-D address space, from 0xE00E0000 to 0xE00E0FFC when accessed by the debugger, and from 0x5C000000 to 0x5C000FFC when accessed from the system bus.

Table 613. System ROM table

Address offset in ROM table	Component name	Component base address (debugger)	Component base address (system bus)	Component address offset	Size	Entry
0x000	DBGMCU	0xE00E1000	0x5C001000	0x01000	4 Kbyte	0x00001003
0x004	-	-	-	-	4 Kbyte	0x00002002
0x008	SWO	0xE00E3000	0x5C003000	0x03000	4 Kbyte	0x00003003
0x00C	System CTI-1	0xE00E4000	0x5C004000	0x04000	4 Kbyte	0x00004003
0x010	System CTI-2	0xE00E5000	0x5C005000	0x05000	4 Kbyte	0x00005003
0x014	Trace Funnel	0xE00E6000	0x5C006000	0x06000	4 Kbyte	0x00006003
0x018	TPIU	0xE00E7000	0x5C007000	0x07000	4 Kbyte	0x00007003
0x01C	Replicator	0xE00E8000	0x5C008000	0x08000	4 Kbyte	0x00008003
0x020	ETF	0xE00E9000	0x5C009000	0x09000	4 Kbyte	0x00009003
0x024 to 0xFC8	Reserved	-	-	-	-	0x00000000
0xFCC to 0xFFC	ROM table registers	-	-	-	-	See *System ROM registers*

The top of each ROM table contains a number of read-only registers, including the standard CoreSight component and peripheral identity registers, see section *System ROM registers*.

Each debug component occupies one or more 4 Kbyte blocks of address space. This block of address space is referred to as the debug register file for the component.

The component address offset field of a ROM Table entry points to the start of the last 4 Kbyte block of the address space of the component. This block always contains the component and peripheral ID registers for the component, starting at offset 0xFD0 from the start of the block. The 4 Kbyte count field PIDR4 [7:4], specifies the number of 4 Kbyte blocks for the component. Therefore, the process for finding the start of the address space for a component is:

1. Read the ROM-table entry for the component and extract its Address_Offset [18:0] from bits [31:12] of the ROM-table entry.
2. Use the address offset, together with the base address of the ROM table, ROM_Base_Address, to calculate the base address of the component:

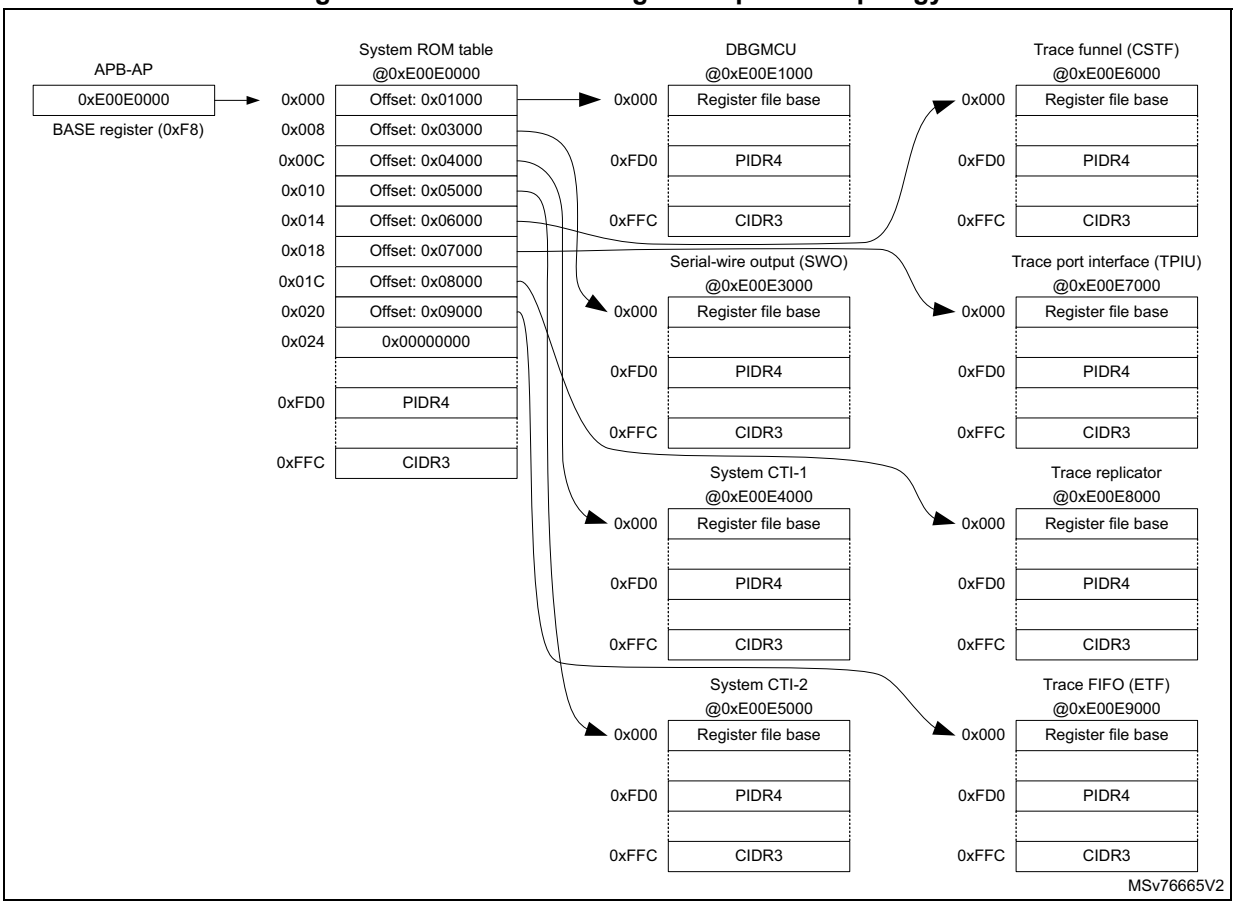
   Component_Base_Address = ROM_Base_Address + Address_Offset

   The Component_Base_Address is the start address of the 4 Kbyte block of the address space for the component.
3. Read the peripheral ID4 register for the component. The address of this register is:

   Peripheral_ID4_address = Component_Base_Address + 0xFD0
4. Extract the 4 Kbyte count field [7:4] from the value of the Peripheral ID4 Register.
5. Use the 4 Kbyte count field value to calculate the start address of the address space for the component. If the field value is 0b0000, which corresponds to a count value of 1, the address space for the component starts at Component_Base_Address obtained at stage 2.

The topology for the CoreSight components on the APB-D is shown in *Figure 1047*.

Figure 1047. APB-D CoreSight component topology



For more information on the use of the ROM table, refer to the Arm® Debug Interface Architecture Specification *[1]*.

52.6.2 System ROM registers

SYSROM memory type register (SYSROM_MEMTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory
0: No system memory is present on this bus

SYSROM CoreSight peripheral identity register 4 (SYSROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC continuation code

SYSROM CoreSight peripheral identity register 0 (SYSROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 007B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Device part number field, bits [7:0]
0x7B: STM32H7Pxxx device

SYSROM CoreSight peripheral identity register 1 (SYSROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Device part number field, bits [11:8]
0x4: STM32H7Pxxx

SYSROM CoreSight peripheral identity register 2 (SYSROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Device revision number
0x1: Rev A

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x2: STMicroelectronics JEDEC code

SYSROM CoreSight peripheral identity register 3 (SYSROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

SYSROM CoreSight component identity register 0 (SYSROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

SYSROM CoreSight component identity register 1 (SYSROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

SYSROM CoreSight component identity register 2 (SYSROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

SYSROM CoreSight component identity register 3 (SYSROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

52.6.3 System ROM register map and reset values

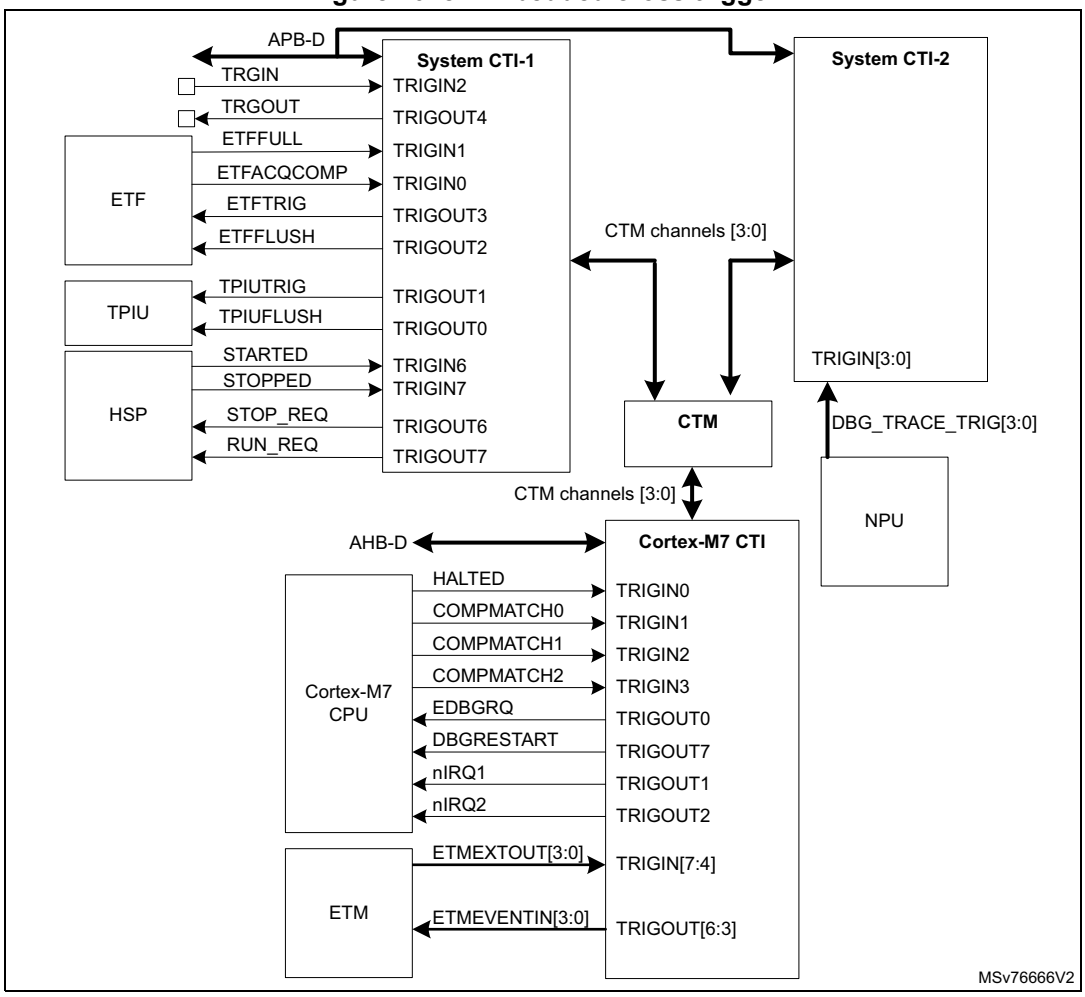
Table 614. System ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	SYSROM_ MEMTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
0xFD0	SYSROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3;0			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	SYSROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	1	1	1	0	1	1
0xFE4	SYSROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	0
0xFE8	SYSROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	1	1	0	1	0
0xFEC	SYSROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0xFF0	SYSROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	1	0	1
0xFF4	SYSROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	1	0	0	0	0
0xFF8	SYSROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	1
0xFFC	SYSROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	1

52.6.4 Cross trigger interfaces (CTI) and matrix (CTM)

The cross trigger interfaces (CTI) and cross trigger matrix (CTM) together form the CoreSight embedded cross trigger feature. There are three CTI components, two at system level and one dedicated to the Cortex-M7. The three CTIs are connected to each other via the CTM. The system-level CTI is accessible to the debugger via the system access port and associated APB-D. The Cortex-M7 CTI is physically integrated in the Cortex-M7 core, and is accessible via the Cortex-M7 access port and associated AHBD.

Figure 1048. Embedded cross trigger



The CTIs allow events from various sources to trigger debug and/or trace activity. For example, a transition detected on an external trigger input can start code trace. It is also possible to send the NPU dbg_trace_trig events to the ETM (thanks to ETMEVENTIN), so these events can appear as trigger packets in the reconstructed trace.

Each CTI has up to 8 trigger inputs and 8 trigger outputs. Any input can be connected to any output, on the same CTI, or on another CTI via the CTM.

The trigger input and output signals for each CTI are listed in *Table 615* to *Table 619*.

Table 615. System CTI inputs

#	Source signal	Source component	Comments
0	ETFACQCOMP	ETF	ETF capture finished - allows a debug event to be generated when the trace FIFO is empty
1	ETFFULL	ETF	ETF full flag - allows a debug event to be generated when the trace FIFO is full
2	DBTRIGI	GPIO	External trigger input - allows an external signal to generate a debug event
3	-	-	Not used
4	-	-	Not used
5	-	-	Not used
6	STARTED	HSP	Asserted when HSP exit from debug state
7	STOPPED	HSP	Asserted when HSP enters in debug state

Table 616. System CTI outputs

#	Output signal	Destination component	Comments
0	TPIUFLUSH	TPIU	Trace port flush trigger - causes the TPIU FIFO to be flushed
1	TPIUTRIG	TPIU	Trace Port enable trigger - starts trace output on the external trace port
2	ETFFLUSH	ETF	ETF flush trigger - causes the Trace FIFO to be flushed
3	ETFTRIG	ETF	ETF enable trigger - starts filling the Trace FIFO
4	DBTRIGO	GPIO	External IO trigger output - allows monitoring of events on the external DBTRIGO pin
5	-	-	Not used
6	STOP_REQ	HSP	Debug Request to HSP
7	RUN_REQ	HSP	Request to exit from debug state

Table 617. System CTI-2 inputs(1)

#	Source signal	Source component	Comments
0	DBG_TRACE_TRIG[0]	NPU	Trigger event from the NPU Debug Trace Unit
1	DBG_TRACE_TRIG[1]	NPU	Trigger event from the NPU Debug Trace Unit
2	DBG_TRACE_TRIG[2]	NPU	Trigger event from the NPU Debug Trace Unit
3	DBG_TRACE_TRIG[3]	NPU	Trigger event from the NPU Debug Trace Unit

Table 617. System CTI-2 inputs(1) (continued)

#	Source signal	Source component	Comments
4	-	-	Not used
5	-	-	Not used
6	-	-	Not used
7	-	-	Not used

1. System CTI-2 outputs are not used.

Table 618. Cortex-M7 CTI inputs

#	Source signal	Source component	Comments
0	HALTED	Cortex-M7 CPU	CPU halted - indicates CPU is in Debug mode
1	COMPMATCH0	Cortex-M7 DWT	DWT comparator 0 match
2	COMPMATCH1	Cortex-M7 DWT	DWT comparator 1 match
3	COMPMATCH2	Cortex-M7 DWT	DWT comparator 2 match
4	ETMEXTOUT0	Cortex-M7 ETM	ETM external trigger out
5	ETMEXTOUT1	Cortex-M7 ETM	ETM external trigger out
6	-	-	Not used
7	-	-	Not used

Table 619. Cortex-M7 CTI outputs

#	Output signal	Destination component	Comments
0	EDBGRQ	Cortex-M7 CPU	CPU halt request - puts CPU in Debug mode
1	nIRQ1	Cortex-M7 NVIC	Interrupt request
2	nIRQ2	Cortex-M7 NVIC	Interrupt request
3	ETMEVENTIN0	Cortex-M7 ETM	ETM trig request - enables CPU execution trace
4	ETMEVENTIN1	Cortex-M7 ETM	ETM trig request - enables CPU execution trace
5	ETMEVENTIN2	Cortex-M7 ETM	ETM trig request - enables CPU execution trace
6	ETMEVENTIN3	Cortex-M7 ETM	ETM trig request - enables CPU execution trace
7	DBGRESTART	Cortex-M7 CPU	CPU restart request - CPU exits Debug mode

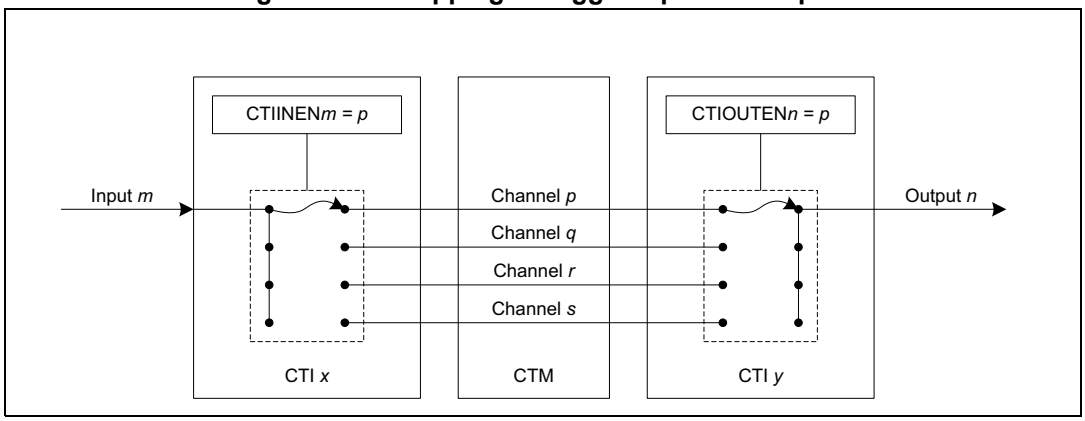
There are four event channels in the cross trigger matrix, which allows up to four parallel bidirectional connections between trigger inputs and outputs on different CTIs. To connect input number *m* on CTI *x* to output number *n* on CTI *y*, the input must be connected to an

event channel *p* using the CTIINEN*m* register of CTI *x*. The same channel *p* must be connected to the output using the CTIOUTEN*n* register of CTI *y*.

Note: This applies even if the input and output belong to the same CTI.

An input can be connected to more than one channel (up to four), so an input can be routed to several outputs. Similarly, an output can be connected to several inputs. It is also possible to connect several inputs/outputs to the same channel.

Figure 1049. Mapping of trigger inputs to outputs



For more information on the cross-trigger interface CoreSight component, refer to the Arm® CoreSight™ SoC-400 Technical Reference Manual *[2]*.

52.6.5 CTI registers

The register file base address for each CTI is defined by the ROM table for the bus to which it is connected. The registers are the same for each CTI.

CTI control register (CTI_CONTROL)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **GLBEN**: Global enable.
0: Cross-triggering disabled
1: Cross-triggering enabled

CTI trigger acknowledge register (CTI_INTACK)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **INTACK[7:0]**: Trigger acknowledge

There is one bit of the register for each CTITRIGOUT output. When a 1 is written to a bit in this register, the corresponding CTITRIGOUT output is acknowledged, causing it to be cleared.

CTI application trigger set register (CTI_APPSET)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPSET[3:0]**: Set channel event

Read:
0bXXX0: Channel 0 event inactive
0bXXX1: Channel 0 event active
0bXX0X: Channel 1 event inactive
0bXX1X: Channel 1 event active
0bX0XX: Channel 2 event inactive
0bX1XX: Channel 2 event active
0b0XXX: Channel 3 event inactive
0b1XXX: Channel 3 event active

Write:
0bXXX0: No effect
0bXXX1: Set event on Channel 0
0bXX0X: No effect
0bXX1X: Set event on Channel 1
0bX0XX: No effect
0bX1XX: Set event on Channel 2
0b0XXX: No effect
0b1XXX: Set event on Channel 3

CTI application trigger clear register (CTI_APPCLEAR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR[3:0]			
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPCLEAR[3:0]**: Clear channel event

0bXXX0: No effect
0bXXX1: Clear event on Channel 0
0bXX0X: No effect
0bXX1X: Clear event on Channel 1
0bX0XX: No effect
0bX1XX: Clear event on Channel 2
0b0XXX: No effect
0b1XXX: Clear event on Channel 3

CTI application pulse register (CTI_APPPULSE)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPPULSE[3:0]			
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPPULSE[3:0]**: Pulse channel event

This register clears itself immediately.
0bXXX0: No effect
0bXXX1: Generate pulse on Channel 0
0bXX0X: No effect
0bXX1X: Generate pulse on Channel 1
0bX0XX: No effect
0bX1XX: Generate pulse on Channel 2
0b0XXX: No effect
0b1XXX: Generate pulse on Channel 3

CTI trigger IN x enable register (CTI_INENx)

Address offset: 0x020 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGINEN[3:0]**: Cross-trigger event enable

Enables or disables a cross-trigger event on each of the four channels when CTITRIGINx is activated (x = 0 to 7).
0bXXX0: Trigger n does not generate events on Channel 0
0bXXX1: Trigger n generates events on Channel 0
0bXX0X: Trigger n does not generate events on Channel 1
0bXX1X: Trigger n generates events on Channel 1
0bX0XX: Trigger n does not generate events on Channel 2
0bX1XX: Trigger n generates events on Channel 2
0b0XXX: Trigger n does not generate events on Channel 3
0b1XXX: Trigger n generates events on Channel 3

CTI trigger OUT x enable register (CTI_OUTENx)

Address offset: 0x0A0 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGOUTEN[3:0]**: Enable trigger upon event

For each channel, the field defines whether an event on that channel generates a trigger on CTITRIGOUTx (x = 0 to 7).
0bXXX0: Channel 0 events do not generate triggers on Trigger output n
0bXXX1: Channel 0 events generate triggers on Trigger output n
0bXX0X: Channel 1 events do not generate triggers on Trigger output n
0bXX1X: Channel 1 events generate triggers on Trigger output n
0bX0XX: Channel 2 events do not generate triggers on Trigger output n
0bX1XX: Channel 2 events generate triggers on Trigger output n
0b0XXX: Channel 3 events do not generate triggers on Trigger output n
0b1XXX: Channel 3 events generate triggers on Trigger output n

CTI trigger IN status register (CTI_TRGISTS)

Address offset: 0x130

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGINSTATUS[7:0]**: Trigger input status

There is one bit of the register for each CTITRIGIN input. When a bit is set to 1 it indicates that the corresponding trigger input is active. When it is set to 0, the corresponding trigger input is inactive.

CTI trigger OUT status register (CTI_TRGOSTS)

Address offset: 0x134

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGOUTSTATUS[7:0]**: Trigger output status

There is one bit of the register for each CTITRIGOUT output. When a bit is set to 1 it indicates that the corresponding trigger output is active. When it is set to 0, the corresponding trigger output is inactive.

CTI channel IN status register (CTI_CHINSTS)

Address offset: 0x138

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHINSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHINSTATUS[3:0]**: Channel input status

There is one bit of the register for each channel input. When a bit is set to 1 it indicates that the corresponding channel input is active. When it is set to 0, the corresponding channel input is inactive.

CTI channel OUT status register (CTI_CHOUTSTS)

Address offset: 0x13C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOUTSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHOUTSTATUS[3:0]**: Channel output status

There is one bit of the register for each channel output. When a bit is set to 1 it indicates that the corresponding channel output is active. When it is set to 0, the corresponding channel output is inactive.

CTI channel gate register (CTI_GATE)

Address offset: 0x140

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **GATEEN[3:0]**: Channel output enable

For each channel, defines whether an event on that channel can propagate over the CTM to other CTIs.
0bXXX0: Channel 0 events do not propagate
0bXXX1: Channel 0 events propagate
0bXX0X: Channel 1 events do not propagate
0bXX1X: Channel 1 events propagate
0bX0XX: Channel 2 events do not propagate
0bX1XX: Channel 2 events propagate
0b0XXX: Channel 3 events do not propagate
0b1XXX: Channel 3 events propagate

CTI claim tag set register (CTI_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits
Write:
0000: No effect
XXX1: Set bit 0
XX1X: Set bit 1
X1XX: Set bit 2
1XXX: Set bit 3
Read:
0xF: Indicates there are four bits in claim tag

CTI claim tag clear register (CTI_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits
Write:
0000: No effect
XXX1: Clear bit 0
XX1X: Clear bit 1
X1XX: Clear bit 2
1XXX: Clear bit 3
Read: Returns current value of claim tag

CTI lock access register (CTI_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: CTI register write access enable

Enables write access to some CTI registers by processor core (debuggers do not need to unlock the component)

0xC5ACCE55: Enable write access

Other values: Disable write access

CTI lock status register (CTI_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the CTI_LAR register

0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock

This bit always returns zero when read by an external debugger.

0: Write access is permitted

1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism

The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.

0: No lock control mechanism exists

1: Lock control mechanism is implemented

CTI authentication status register (CTI_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x2: Disabled
0x3: Enabled

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x2: Disabled
0x3: Enabled

CTI device configuration register (CTI_DEVID)

Address offset: 0xFC8

Reset value: 0x0004 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]			
												r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMTRIG[7:0]								Res.	Res.	Res.	EXTMUXNUM[4:0]				
r	r	r	r	r	r	r	r				r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **NUMCH[3:0]**: Number of ECT channels available
0x4: 4 channels

Bits 15:8 **NUMTRIG[7:0]**: Number of ECT triggers available
0x8: 8 trigger inputs and 8 trigger outputs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **EXTMUXNUM[4:0]**: Number of trigger input/output multiplexers
0x0: None

CTI device type identifier register (CTI_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification
0x1: Indicates that this component is a cross-triggering component.

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x4: Indicates that this component allows a debugger to control other components in a CoreSight SoC-400 system.

CTI CoreSight peripheral identity register 4 (CTI_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

CTI CoreSight peripheral identity register 0 (CTI_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0006

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x06: CTI part number

CTI CoreSight peripheral identity register 1 (CTI_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: CTI part number

CTI CoreSight peripheral identity register 2 (CTI_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 005B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x5: r1p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

CTI CoreSight peripheral identity register 3 (CTI_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

CTI CoreSight component identity register 0 (CTI_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

CTI CoreSight component identity register 1 (CTI_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

CTI CoreSight component identity register 2 (CTI_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

CTI CoreSight component identity register 3 (CTI_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

52.6.6 CTI register map and reset values

Table 620. CTI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CTI_CONTROL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
0x010	CTI_INTACK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x014	CTI_APPSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
0x018	CTI_APPCLEAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
0x01C	CTI_APPPULSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPPULSE[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
0x020 to 0x03C	CTI_INEN0 to CTI_INEN7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
0x040 to 0x09C	Reserved																																
0x0A0 to 0x0BC	CTI_OUTEN0 to CTI_OUTEN7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
0x0C0 to 0x12C	Reserved	Reserved																															
0x130	CTI_TRIGISTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x134	CTI_TRIGOSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x138	CTI_CHINSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHISTATUS[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
0x13C	CTI_CHOUTSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOSTATUS[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
0x140	CTI_GATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1
0x144 to 0xF9C	Reserved																																
0xFA0	CTI_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1
0xFA4	CTI_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0

Table 620. CTI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFA8 to 0xFAC	Reserved																																
0xFB0	CTI_LAR	ACCESS_W[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xFB4	CTI_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	1
0xFB8	CTI_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	0	1	0
0xFC8	CTI_DEVID	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NUMCH[3:0]				NUMTRIG[7:0]								Res	Res	Res	EXMUXNUM [4:0]				
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	0	1	0	0	0	0	0	0	1	0	0	0	-	-	-	0	0	0	0	0
0xFCC	CTI_DEVTYPE	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SUBTYPE[3:0]				MAJORTYPE [3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	1	0	1	0	0
0xFD0	CTI_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	CTI_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	1	0
0xFE4	CTI_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	1	0	0	1
0xFE8	CTI_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID [6:4]		
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	0	1	1	0	1	1
0xFEC	CTI_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0xFF0	CTI_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	1	0	1
0xFF4	CTI_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	0	1	0	0	0	0
0xFF8	CTI_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	1
0xFFC	CTI_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	1

52.6.7 Trace funnel (CSTF)

The trace funnel is a CoreSight component that combines the ATB buses from two trace sources into one single ATB. The CSTF has two ATB slave ports, and one ATB master port. An arbiter selects the slave ports according to a programmable priority.

The slave ports are connected as follows:

- S0: Cortex-M7 ETM
- S1: Cortex-M7 ITM

The CSTF registers allow the slave ports to be individually enabled, and their priority settings to be configured. The priorities can be modified only when the trace is disabled. The arbitration works as follows:

- The arbiter selects the slave port with the highest assigned priority that has valid data
- Transfers are passed from the selected slave to the master port until *min_hold_time is reached*, where *min_hold_time* is programmable in the CONTROL register.
- A new arbitration is then performed.

High priority should be assigned to slave ports connected to sources with a small amount of buffering, or where data loss cannot be tolerated. Low priority should be assigned to less critical sources or those with large buffers.

For more information on the ATB Funnel CoreSight component, refer to the Arm® CoreSight™ SoC-400 Technical Reference Manual *[2]*.

52.6.8 Trace funnel registers

CSTF control register (CSTF_CTRL)

Address offset: 0x000

Reset value: 0x0000 0300

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MIN_HOLD_TIME[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	ENS1	ENS0
				rw	rw	rw	rw							rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **MIN_HOLD_TIME[3:0]**: Number of transactions between arbitrations.
0x0: 1 transaction
:
0xE: 15 transactions
0xF: Reserved

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **ENS1**: S1 slave port enable
0: Disable port
1: Enable port

Bit 0 **ENS0**: S0 slave port enable
0: Disable port
1: Enable port

CSTF priority register (CSTF_PRIORITY)

Address offset: 0x004

Reset value: 0x0000 0688

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIPORT1[2:0]			PRIPORT0[2:0]		
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:3 **PRIPORT1[2:0]**: S1 slave port priority
0: Highest priority
:
7: Lowest priority

Bits 2:0 **PRIPORT0[2:0]**: S0 slave port priority
0: Highest priority
:
7: Lowest priority

CSTF claim tag set register (CSTF_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits
Write:
0000: No effect
XXX1: Set bit 0
XX1X: Set bit 1
X1XX: Set bit 2
1XXX: Set bit 3
Read:
0xF: Indicates there are four bits in claim tag

CSTF claim tag clear register (CSTF_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits
Write:
0000: No effect
XXX1: Clear bit 0
XX1X: Clear bit 1
X1XX: Clear bit 2
1XXX: Clear bit 3
Read: Returns current value of claim tag

CSTF lock access register (CSTF_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: CSTF register write access enable
The field enables write access to some CSTF registers by the processor cores (debuggers do not need to unlock the component).
0xC5ACCE55: Enable write access
Other values: Disable write access

CSTF lock status register (CSTF_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the CSTF_LAR register
0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock
This bit always returns zero when read by an external debugger.
0: Write access is permitted
1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.
0: No lock control mechanism exists
1: Lock control mechanism is implemented

CSTF authentication status register (CSTF_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x2: Disabled
0x3: Enabled

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x2: Disabled
0x3: Enabled

CSTF CoreSight device identity register (CSTF_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 0024

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCHEME[3:0]				PORTCNT[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SCHEME[3:0]**: Priority scheme
0x2: Static priority

Bits 3:0 **PORTCNT[3:0]**: Number of input ports connected
0x4: Four input ports

CSTF CoreSight device type identity register (CSTF_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVTYPEID[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DEVTYPEID[7:0]**: Device type identifier
0x12: Trace funnel

CSTF CoreSight peripheral identity register 4 (CSTF_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

CSTF CoreSight peripheral identity register 0 (CSTF_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x08: CSTF part number

CSTF CoreSight peripheral identity register 1 (CSTF_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: CSTF part number

CSTF CoreSight peripheral identity register 2 (CSTF_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 003B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x3: r1p1

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

CSTF CoreSight peripheral identity register 3 (CSTF_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

CSTF CoreSight component identity register 0 (CSTF_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

CSTF CoreSight component identity register 1 (CSTF_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

CSTF CoreSight component identity register 2 (CSTF_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

CSTF CoreSight component identity register 3 (CSTF_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

52.6.9 Trace funnel register map and reset values

Table 621. CSTF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CSTF_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIN_HOLD_TIME[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	ENS1	ENS0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	1	1	-	-	-	-	-	-	0	0
0x004	CSTF_PRIORITY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIPORT1[2:0]			PRIPORT0[2:0]		
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	1	0	0	0
0x008 to 0xF9C	Reserved																																
0xFA0	CSTF_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1
0xFA4	CSTF_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
0xFA8to 0xFAC	Reserved																																
0xFB0	CSTF_LAR	ACCESS_W[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xFB4	CSTF_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCTYPE	LOCKGRANT	LOCKEXIST
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	1
0xFB8	CSTF_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	0	1	0

Table 621. CSTF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFC8	CSTF_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCHEME[3:0]				PORTCNT[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	1	0	0	1	0	0
0xFCC	CSTF_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVTYPEID[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	1	0	0	1	0
0xFD0	CSTF_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	CSTF_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	0	0	0
0xFE4	CSTF_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	1	0	0	1
0xFE8	CSTF_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	1	1	1	0	1	1
0xFEC	CSTF_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0xFF0	CSTF_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	1	0	1
0xFF4	CSTF_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	0	1	0	0	0	0
0xFF8	CSTF_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	1
0xFFC	CSTF_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	1

52.6.10 Embedded trace FIFO (ETF)

The ETF is a 2 Kbyte memory that captures trace data from two trace sources, namely the ETM and ITM of the CPU core. The ETF is a design configuration of the CoreSight™ trace memory controller component.

The ETF can be used in three modes (selected in the mode register):

1. Hardware FIFO mode

   The trace memory is used as a FIFO that is drained through the ATB master interface. Trace data is captured by the trace RAM and when full, the incoming trace stream is stalled. When the Trace buffer is not empty, trace data is drained out through the ATB master interface to the TPIU.

   In this mode, the role of the FIFO is to smooth the flow of trace information arriving at the trace port. Since the trace data can be very bursty by nature, the peak data rate can easily exceed the port capability, resulting in an overflow. The ETF allows a steady data rate at the trace port, which can then be sized according to the average rate rather than the peak. The trace is stored off-chip in real time by the trace port analyzer tool, and so the trace log can be very big.

2. Software FIFO mode

   The trace memory is used as a FIFO that can be read through the RRD Register while the trace is being captured. Trace data is captured by the trace RAM and when full, the incoming trace stream is stalled.

   This mode allows the trace to be transferred by DMA into the system memory, or to a high-speed interface (SPI, USB and so on), or even monitored by software. Note that unlike the hardware FIFO mode, this mode is invasive, since it uses system resources which are shared by the processor.

3. Circular buffer mode

   The trace memory is used as a circular buffer. Trace data is captured in the trace memory starting from the location pointed to by the write pointer register. When the trace memory is full, incoming trace data continues to be overwritten in the trace memory until a stop condition occurs.

   In this mode, the ETF stores the trace data on-chip, so the trace log size is limited to that of the ETF SRAM, 2 Kbytes in this case. Being a circular buffer, when the FIFO is full, incoming trace data overwrites the oldest stored data and the oldest stored data is lost. Therefore the content of the trace buffer represents the most recent activity of the processor, up to the point when the buffer was stopped, rather than all the activity since the trace was started.

   There are three possible methods to read the buffer contents once the trace stops:

   – Via the Trace port - with the TPIU enabled, the contents of the buffer are output over the Trace port. This can be done by setting the DRAINBUF bit in the ETF_FFCR register.
   – Via the Debug port - the debugger can read the buffer via the RRD register that is accessible over the system APB-D.
   – By software - the processor can read the buffer via the RRD register, since the APB-D is accessible from the system bus.

The ETF can be moved to any one of these states:

- **Disabled**

  This state is entered after a reset, or when trace capture is disabled. The ETF must only be programmed in this state.
- **Running**

  Trace capture is performed in this state. It is entered by enabling trace capture while in Disabled state.
- **Stopped**

  Trace capture is stopped in this state, but the contents of the buffer can be read out or drained. This state is entered after a stop event (trigger or flush).
- **Disabling**

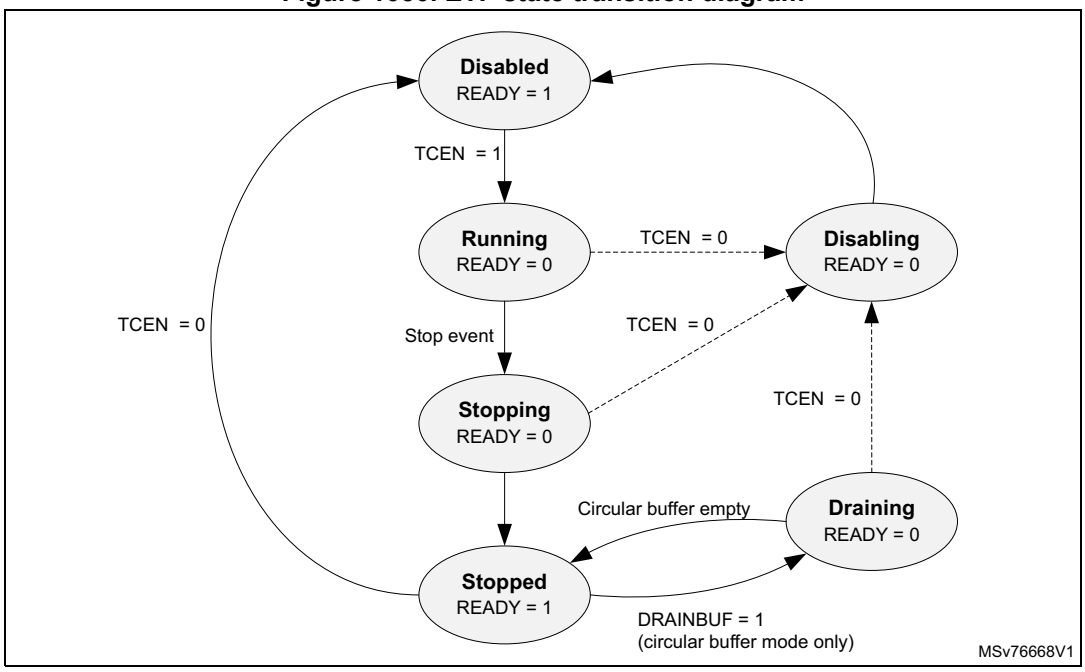
  This is a transition state while disabling trace capture.
- **Stopping**

  This is a transition state while stopping trace capture.
- **Draining**

  This state is entered while draining the buffer in Stopped state.

The state transition diagram is shown in *Figure 1050*.

Figure 1050. ETF state transition diagram



For more information on the CoreSight™ trace memory controller component, refer to the Arm® CoreSight™ trace memory controller technical reference manual *[3]*.

52.6.11 ETF registers

ETF RAM size register (ETF_RSZ)

Address offset: 0x004

Reset value: 0x0000 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RSZ[30:16]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSZ[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **RSZ[30:0]**: RAM size

The value of the field indicates the number of 32-bit words
0x200: 512 words = 2 Kbytes

ETF status register (ETF_STS)

Address offset: 0x00C

Reset value: 0x0000 001C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY	FTEMPTY	READY	TRIGD	FULL
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **EMPTY**: Trace FIFO empty

This bit is valid only when the TCEN bit of the ETF_CTL register is high. This bit reads as zero when TCEN is low.

0: Trace FIFO contains data

1: Trace FIFO is empty.

Note: Empty trace FIFO does not mean that the ETF pipeline is empty. The latter is indicated by the FTEMPTY bit.

Bit 3 **FTEMPTY**: Formatter empty

This bit is set when a trace capture has stopped, and all internal pipelines and buffers have been drained. Unlike READY, it is not affected by buffer drains. The ACQCOMP output reflects the value of this bit.

Bit 2 **READY**: ETF ready

This bit is set when a trace capture has stopped and all internal pipelines and buffers have been drained (Stopped or Disabled state).

Bit 1 **TRIGD**: Triggered

The Triggered bit is set when a trace capture is in progress and the TMC has detected a Trigger Event. This bit is cleared when leaving Disabled state.

This bit is operational only in the Circular buffer mode. In all other modes, this bit is always low.

This bit does not indicate that a trigger has been embedded in the formatted output trace data from the TMC. Trigger indication on the output trace stream is determined by the programming of the Formatter and Flush Control Register, ETF_FFCR.

Bit 0 **FULL**: Trace buffer full

In circular buffer mode, this flag is set when the RAM write pointer wraps around the top of the buffer, and remains set until the TCEN bit of the ETF_CTL register is cleared and set.

In software and hardware FIFO modes, this flag indicates that the current space in the trace memory is less than or equal to the value programmed in the ETF_BUFWM Register, that is, Fill level >= MEM_SIZE - BUFWM.

This bit is cleared when leaving Disabled state. The FULL output reflects the value of this register bit.

ETF RAM read data register (ETF_RRD)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RRD[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RRD[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RRD[31:0]**: RAM Read Data.

Circular buffer mode:

When in Stopped state and the buffer is not empty, reading this register returns the next word of data from the trace buffer. When the whole trace buffer has been read, the Empty bit in the ETF_STS Register is set, and subsequent reads return 0xFFFFFFF. Reading this register when not in Stopped state returns 0xFFFFFFFF.

Software FIFO mode:

Reading this register returns data from the FIFO. If this register is read when the FIFO is empty, the data returned is 0xFFFFFFFF.

Hardware FIFO mode:

Reading this register returns 0xFFFFFFFF.

ETF RAM read pointer register (ETF_RRP)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RRP[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **RRP[12:0]**: RAM Read Pointer

The RAM Read Pointer Register contains the value of the read pointer that is used to read entries from the trace memory over the APB interface via the ETF_RRD register. The pointer can be programmed with a byte address, 64-bit aligned (that is, bits 0 to 3 should be zero). The pointer is incremented by 8 each time a full 64-bit FIFO entry has been written. When the pointer reaches its maximum value, it wraps around.

This register can only be written to while in Disabled state. It can be read in Disabled state, in Stopped state in Circular buffer mode and SW FIFO mode, and also in Running and Stopping states in SW FIFO mode.

ETF RAM write pointer register (ETF_RWP)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RWP[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **RWP[12:0]**: RAM write pointer

The RAM write pointer register contains the value of the write pointer that is used to write entries into the trace memory over the APB interface via the ETF_RWD register. The pointer can be programmed with a byte address, 64-bit aligned (that is, bits 0 to 3 should be zero). The pointer is incremented by 8 each time a full 64-bit FIFO entry has been read. When the pointer reaches its maximum value, it wraps around.

This register can only be written to while in Disabled state. It can be read in Disabled state, in Stopped state in Circular buffer mode and SW FIFO mode, and also in Running and Stopping states in SW FIFO mode.

ETF trigger counter register (ETF_TRG)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TRG[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **TRG[10:0]**: Trigger counter

In Circular buffer mode, specifies the number of 32-bit words to capture in the trace RAM following the detection of either a rising edge on the TRIGIN input or a trigger packet in the incoming trace stream, ATID = 7'h7D. On capturing the specified number of data words, a trigger event occurs. The effect of a trigger event on the ETF behavior is controlled by the FFCR Register.

The number of 32-bit words written into the trace RAM following the trigger is the value stored in this register, plus one. This register is ignored when the ETF is in Software FIFO mode or Hardware FIFO mode. When the trigger counter starts counting, any additional triggers, either on TRIGIN or in the incoming trace stream, are ignored until the counter reaches zero. When the trigger counter has reached zero, it remains at zero until it is re-programmed with a write to this register.

This register is cleared when READY goes high, so that the state of the counter when trace capture has stopped does not affect a subsequent trace capture session. Writing to this register when not in Disabled state results in unpredictable behavior.

A read access to this register is permitted at any time when in Disabled state, or in Circular buffer mode. A read access returns the current value of the trigger counter.

ETF control register (ETF_CTL)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **TCEN**: Trace capture enable

When writing:

0: Disable trace capture (moves from Running, Stopping or Stopped state into Disabling or Disabled state)

1: Enable trace capture (moves from Disabled state to Running state)

When reading, this bit is low when in Disabling or Disabled states, and high otherwise.

ETF RAM write data register (ETF_RWD)

Address offset: 0x024

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RWD[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RWD[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **RWD[31:0]**: RAM write data

When in Disabled state, a write to this register stores data at the location pointed to by the RWP. Writes to this register when not in Disabled state are ignored. When a full memory width (64-bit) of data has been written, the data is written to memory and the RAM Write Pointer is incremented to the next memory word.

This register is used for test purposes.

ETF mode register (ETF_MODE)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **MODE[1:0]**: Operation mode

00b: Circular buffer mode

In this mode, the trace memory is used as a circular buffer. Trace data is captured in the trace memory starting from the location pointed to by the write pointer register. Even when the trace memory is full, incoming trace data continues to be overwritten into the trace memory until a stop condition has occurred.

01b: Software FIFO mode

In this mode, the trace memory is used as a FIFO that can be read through the RRD Register while a trace is being captured. Trace data is captured in the trace RAM and when full, the incoming trace stream is stalled.

10b: Hardware FIFO mode

In this mode, the trace memory is used as a FIFO that is drained through the ATB master interface. Trace data is captured in the trace RAM and when full, the incoming trace stream is stalled. When the trace buffer is non-empty, trace data is drained out through the ATB master interface to the TPIU.

11b: Reserved

ETF latched buffer fill level register (ETF_LBUFLVL)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LBUFLEVEL[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **LBUFLEVEL[11:0]**: Latched buffer fill level

Reading this register returns the maximum fill level of the trace memory in 32-bit words since this register was last read. Reading this register also results in its contents being updated to the current fill level.

When entering Disabled state, this register retains its last value. While in Disabled state, reads from this register do not affect its value. When exiting Disabled state, the LBUFLEVEL register is cleared.

This register is used for performance analysis of the trace system.

ETF current buffer fill level register (ETF_CBUFLVL)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CBUFLEVEL[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **CBUFLEVEL[11:0]**: Current buffer fill level

Reading this register returns the current fill level of the trace memory in 32-bit words.

This register is cleared when TCEN is low.

ETF buffer level watermark register (ETF_BUFWM)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BUFWM[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **BUFWM[10:0]**: Buffer level watermark

The value programmed into this register indicates the required threshold vacancy level of the trace memory in 32-bit words . When the space in the FIFO is less than or equal to this value, that is, fill level >= MEM_SIZE - BUFWM, the FULL output is pulled high and the FULL bit in the STS Register is set.

This register is used only in Software FIFO and Hardware FIFO modes. In Circular buffer mode, this functionality can be obtained by programming the RWP to the required vacancy trigger level, so that when the pointer wraps around, the FULL bit is set indicating that the vacancy level has fallen below the required level.

The maximum value that can be written into this register is MEM_SIZE - 1. In this case, the FULL bit output is asserted after the first 32-bit word is written to trace memory.

Writing to this register other than when in disabled state results in unpredictable behavior.

ETF formatter and flush status register (ETF_FFSR)

Address offset: 0x300

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTSTOPPED	FLINPROG
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **FTSTOPPED**: Formatter stopped

This bit behaves in the same way as the FTEMPTY bit in the ETF_STS register.

Bit 0 **FLINPROG**: Flush in progress

Indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the ETF_FFCR register, or requested by the ATB master port.

0: No flush in progress

1: Flush in progress

ETF formatter and flush control register (ETF_FFCR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DRAINBUF	STPONTRGEV	STOPONFL	Res.	TRIGONFL	TRGONTRGEV	TRGONTRGIN	Res.	FLUSHMAN	FONTRGEV	FONFLIN	Res.	Res.	ENTI	ENFT
	rw	rw	rw		rw	rw	rw		rw	rw	rw			rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **DRAINBUF**: Drain buffer

This bit is used to enable draining of the trace data through the ATB master interface after the formatter has stopped. This is useful in circular buffer mode to capture trace data into trace memory and then to drain the captured trace through the ATB master interface.

Writing a 1 to this bit when in Stopped state starts the drain of the trace buffer contents through the ATB Master interface. This bit always reads as zero. The READY bit in the ETF_STS register goes low while the drain is in progress.

This bit is only functional when the ETF is in Circular buffer mode and formatting is enabled, that is, the ENFT bit in the ETF_FFCR register is set. Setting this bit when the ETF is in any other mode, or when not in Stopped state, results in Unpredictable behavior.

When trace capture is complete in Circular buffer mode, all of the captured trace must be retrieved from the trace memory through the same mechanism, either read all trace data out through RRD reads, or drain all trace data by setting the DRAINBUF bit. Setting the DRAINBUF bit after some of the captured trace has been read out through RRD results in unpredictable behavior.

Bit 13 **STPONTRGEV**: Stop on trigger event

0: No effect

1: Stop trace capture when a trigger event occurs

Enabling the ETF in Software FIFO mode or Hardware FIFO mode with this bit set results in unpredictable behavior.

Bit 12 **STOPONFL**: Stop on flush

0: No effect

1: Stop trace capture when flush is completed

If a flush is initiated by the ATB master interface, its completion does not lead to a formatter stop regardless of the value programmed in this bit.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **TRIGONFL**: Trigger on flush

0: No effect

1: Indicates a trigger in the trace stream when flush is completed

If ENFT and ENTI are both clear, this bit is ignored and no trigger is inserted into the trace stream.

If a flush is initiated by the ATB master interface, its completion does not lead to a trigger indication regardless of the value programmed in this bit.

Bit 9 **TRGONTRGEV**: Trigger on trigger event

0: No effect

1: Indicates a trigger in the trace stream when a trigger event occurs

If ENFT and ENTI are both clear, this bit is ignored and no trigger is inserted into the trace stream.

This bit is not supported in Software FIFO mode nor Hardware FIFO mode.

Bit 8 **TRGONTRGIN**: Trigger on trigger in

0: No effect

1: Indicates a trigger in the trace stream when a rising edge is detected on the TRIGIN input.

If ENFT and ENTI are both clear, this bit is ignored and no trigger is inserted into the trace stream.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FLUSHMAN**: Manual flush

0: No effect

1: Flush the trace FIFO and pipeline

This bit is cleared automatically when the flush is complete. If the TCEN bit in the ETF_CTL register is 0, writes to this bit are ignored.

Bit 5 **FONTRGEV**: Flush on trigger event

0: No effect

1: Flush the trace FIFO and pipeline if a trigger event occurs

This bit is not supported in Software FIFO mode or Hardware FIFO mode. If STPONTRGEV is set, this bit is ignored.

Bit 4 **FONFLIN**: Flush on flush in

0: No effect

1: Flush the trace FIFO and pipeline when a rising edge is detected on the FLUSHIN input

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **ENTI**: Enable trigger insertion

Setting this bit enables the insertion of triggers in the formatted trace stream. A trigger is indicated by inserting one byte of data 8'h00 with ATID 7'h7D in the trace stream. Trigger indication on the trace stream is further controlled by the register bits TRIGONFL, TRGONTRGEV, and TRGONTRGIN in the FFCR Register. This bit can only be changed when READY is high, and TCEN is low. This bit takes effect only when the ENFT register bit in this register is set. If ENTI bit is set to high when ENFT is low, it results in formatting being enabled.

Bit 0 **ENFT**: Enable formatting.

0: Formatting is disabled. Incoming trace data is assumed to be from a single trace source.

1: Formatting is enabled.

If multiple ATIDs are received by the ETF when trace capture is enabled and the formatter is disabled, an interleaving of trace data occurs. Disabling of formatting is supported only in Circular buffer mode. If the ETF is enabled in a mode other than Circular buffer mode with ENFT low, then formatting is enabled. If ENTI bit is set to high when ENFT is low, formatting is enabled.

This bit is ignored when in Disabled state.

ETF periodic synchronization counter register (ETF_PSCR)

Address offset: 0x308

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSCOUNT[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PSCOUNT[4:0]**: Synchronization counter reload value

Determines the reload value of the Synchronization Counter. The reload value takes effect the next time the counter reaches zero. Reads from this register return the reload value programmed into this register. This register is set to 0xA on reset, corresponding to a synchronization period of 1024 bytes.

0x0: Synchronization disabled

0x1-0x6: Reserved

0x7-0x1B: Synchronization period is $2^{PSCOUNT}$ bytes

0x1C-0x1F: Reserved

ETF claim tag set register (ETF_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:

0000: No effect

XXX1: Set bit 0

XX1X: Set bit 1

X1XX: Set bit 2

1XXX: Set bit 3

Read:

0xF: Indicates there are four bits in claim tag

ETF claim tag clear register (ETF_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits

Write:
0000: No effect
XXX1: Clear bit 0
XX1X: Clear bit 1
X1XX: Clear bit 2
1XXX: Clear bit 3
Read: Returns current value of claim tag

ETF lock access register (ETF_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: ETF register access enable

Enables write access to some ETF registers by the processor cores (debuggers do not need to unlock the component)
0xC5ACCE55: Enable write access
Other values: Disable write access

ETF lock status register (ETF_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the ETF_LAR register
0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock
This bit always returns zero when read by an external debugger.
0: Write access is permitted
1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.
0: No lock control mechanism exists
1: Lock control mechanism is implemented

ETF authentication status register (ETF_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x0: Not implemented

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x0: Not implemented

ETF device configuration register (ETF_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 01C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	MEMWIDTH[2:0]			CONFIGTYP[1:0]		CLKSCHEM	ATBINPORTCNT[4:0]				
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **MEMWIDTH[2:0]**: Memory interface data bus width
0x3: 64 bits (corresponds to 32-bit ATB data)

Bits 7:6 **CONFIGTYP[1:0]**: Configuration type of component (ETB, ETR or ETF)
0x2: ETF

Bit 5 **CLKSCHEM**: RAM clocking scheme (synchronous or asynchronous)
0: Synchronous

Bits 4:0 **ATBINPORTCNT**[4:0]: Number/type of ATB input port multiplexing
0x0: None

ETF device type identifier register (ETF_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0032

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification
0x3: Captures the ATB slave interface trace data in RAM which can then be drained through the ATB master interface

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x2: The component is a trace link because it has an ATB master interface through which trace data can be drained out in Hardware FIFO mode.

ETF CoreSight peripheral identity register 4 (ETF_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

ETF CoreSight peripheral identity register 0 (ETF_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0061

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x61: ETF part number

ETF CoreSight peripheral identity register 1 (ETF_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: ETF part number

ETF CoreSight peripheral identity register 2 (ETF_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x1: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

ETF CoreSight peripheral identity register 3 (ETF_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

ETF CoreSight component identity register 0 (ETF_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

ETF CoreSight component identity register 1 (ETF_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

ETF CoreSight component identity register 2 (ETF_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

ETF CoreSight component identity register 3 (ETF_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

52.6.12 ETF register map and reset values

Table 622. ETF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	ETF_RSZ	Res.	RSZ[30:0]																														
	Reset value	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x008	Reserved																																
0x00C	ETF_STS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY	FTEMPTY	READY	TRIGD	FULL
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	0	0
0x010	ETF_RRD	RRD[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x014	ETF_RRP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RRP[12:0]												
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	ETF_RWP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWP[12:0]												
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	ETF_TRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRG[10:0]										
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0
0x020	ETF_CTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCEN
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
0x024	ETF_RWD	RWD[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x028	ETF_MODE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[1:0]	
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0
0x02C	ETF_LBUFLVL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LBUFLEVEL[11:0]											
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0
0x030	ETF_CBUFLVL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBUFLEVEL[11:0]											
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0
0x034	ETF_BUFWM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUFWM[10:0]										
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0
0x038 to 0x2FC	Reserved																																
0x300	ETF_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTSTOPPED	FLINPROG
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0

Table 622. ETF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x304	ETF_FFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRAINBUF	STPONTRGEV	STOPONFL	Res.	TRGONFL	TRGONTRGEV	TRGONTRGIN	Res.	FLUSHMAN	FONTRGEV	FONFLIN	Res.	Res.	ENTI	ENFT
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	-	0	0	0	-	0	0	0	-	-	0	0
0x308	ETF_PSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSCOUNT[4:0]				
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	0	1	0
0x30C to 0xF9C	Reserved																																
0xFA0	ETF_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1
0xFA4	ETF_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
0xFA8 to 0xFAC	Reserved																																
0xFB0	ETF_LAR	ACCESS_W[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xFB4	ETF_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	1
0xFB8	ETF_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0xFBC to 0xFC4	Reserved																																
0xFC8	ETF_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEMWDTH[2:0]			CONFIGTYP[1:0]		CLKSCHEM	ATBINPORTCNT[4:0]				
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	1	1	1	0	0	0	0	0	0

Table 622. ETF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	ETF_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	1	1	0	0	1	0
0xFD0	ETF_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	ETF_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	1	0	0	0	0	1
0xFE4	ETF_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	1	0	0	1
0xFE8	ETF_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	1	1	0	1	1
0xFEC	ETF_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0xFF0	ETF_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	1	0	1
0xFF4	ETF_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	0	1	0	0	0	0
0xFF8	ETF_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	1
0xFFC	ETF_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	1

52.6.13 Trace port interface unit (TPIU)

The TPIU is a CoreSight™ component that formats the trace stream and outputs it on the external trace port signals. The TPIU has a single ATB slave port for incoming trace data. The trace port is a synchronous parallel port, comprising a clock output, TRACECK, and four data outputs, TRACED(3:0). The trace port width is programmable in the range 1 to 4. Using a smaller port width reduces the number of test points/connector pins needed, and frees up IOs for other purposes. However it restricts the bandwidth of the trace port and hence the quantity of trace information that can be output in real time. The ATB trace bus must be enabled by setting the TRACECLKEN bit in the DBGMCU_CR register before a trace is sent to the TPIU.

Trace data is output on TRACED[3:0] synchronously with the rising and falling edges of TRACECK. TRACECK is synchronous with the TRACECLKIN input, at half the frequency.

This means that TRACECK is asynchronous with respect to the ATB clock. The asynchronism is handled by a FIFO in the TPIU, which also regulates the flow of data from the ETF. If the trace port bandwidth is not sufficient to handle the amount of trace data, the ETF can overflow and trace data can be missed. In this case the TRACECLKIN frequency can be increased, or the port size increased, up to a maximum of four pins. Alternatively, the ETM can be programmed to filter unnecessary information.

For more information on the Trace port interface CoreSight™ component, refer to the Arm® CoreSight™ SoC-400 technical reference manual *[2]*.

52.6.14 TPIU registers

TPIU supported port size register (TPIU_SUPPSIZE)

Address offset: 0x000

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PORTSIZE[31:0]**: Indicates supported trace port sizes, from 1 to 32 pins
Bit n-1 when set indicates that port size n is supported.
0x0000 000F: Port sizes 1 to 4 supported

TPIU current port size register (TPIU_CURPSIZE)

Address offset: 0x004

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PORTSIZE[31:0]**: Current trace port size
Setting bit n-1 indicates that the current port size is n-pin wide. The value of n must be within the range of supported port sizes (1-4). Only one bit can be set, or unpredictable behavior may result. This register should only be modified when the formatter is stopped.

TPIU supported trigger modes register (TPIU_SUPTRGM)

Address offset: 0x100

Reset value: 0x0000 011F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGRUN	TRIGD
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCOUNT8	Res.	Res.	Res.	MULT64K	MULT256	MULT16	MULT4	MULT2
							r				r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **TRGRUN**: Trigger running
0: Trigger has not occurred or counter is at 0
1: Trigger has occurred and counter is not at 0

Bit 16 **TRIGD**: Triggered
0: Trigger has not occurred
1: Trigger has occurred and counter has reached 0

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **TCOUNT8**: 8-bit counter register
1: Implemented

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **MULT64K**: Multiplying the trigger counter by 65536 support
1: Supported

Bit 3 **MULT256**: Multiplying the trigger counter by 256 support
1: Supported

Bit 2 **MULT16**: Multiplying the trigger counter by 16 support
1: Supported

Bit 1 **MULT4**: Multiplying the trigger counter by 4 support
1: Supported

Bit 0 **MULT2**: Multiplying the trigger counter by 2 support
1: Supported

TPIU trigger counter value register (TPIU_TRGCNT)

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGCOUNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGCOUNT[7:0]**: Enable trigger delay indication

Enables delaying the indication of triggers to any externally connected trace capture or storage devices. This counter is only eight bits wide and is intended to be used only with the counter multipliers in the Trigger multiplier register, 0x108. When a trigger is started, this value, in conjunction with the multiplier, specifies the number of words before the trigger is indicated. When the trigger counter reaches 0, the value written here is reloaded. Writing to this register causes the trigger counter value to reset but does not reset any value on the multiplier. Reading this register returns the preset value, not the current count.

TPIU trigger multiplier register (TPIU_TRGMULT)

Address offset: 0x108

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MULT64K	MULT256	MULT16	MULT4	MULT2
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **MULT64K**: Multiply the trigger counter by 65536
0: Disabled
1: Enabled

Bit 3 **MULT256**: Multiply the trigger counter by 256
0: Disabled
1: Enabled

Bit 2 **MULT16**: Multiply the trigger counter by 16
0: Disabled
1: Enabled

Bit 1 **MULT4**: Multiply the trigger counter by 4
0: Disabled
1: Enabled

Bit 0 **MULT2**: Multiply the trigger counter by 2
0: Disabled
1: Enabled

TPIU supported test patterns/modes register (TPIU_SUPTPM)

Address offset: 0x200

Reset value: 0x0003 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCONTEN	PTIMEEN
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATF0	PATA5	PATW0	PATW1
												r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **PCONTEN**: Continuous mode support
1: Supported

Bit 16 **PTIMEEN**: Timed mode support
1: Supported

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **PATF0**: Support of FF/00 pattern
Indicates whether the FF/00 pattern is supported as output over the trace port.
1: Supported

Bit 2 **PATA5**: Support of AA/55 pattern
Indicates whether the AA/55 pattern is supported as output over the trace port.
1: Supported

Bit 1 **PATW0**: Support of walking 0's pattern
Indicates whether the walking 0's pattern is supported as output over the trace port.
1: Supported

Bit 0 **PATW1**: Support of walking 1's pattern
Indicates whether the walking 1's pattern is supported as output over the trace port.
1: Supported

TPIU current test pattern/mode register (TPIU_CURTPM)

Address offset: 0x204

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCONTEN	PTIMEEN
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATF0	PATA5	PATW0	PATW1
												rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **PCONTEN**: Continuous mode enable
0: Disabled
1: Enabled

Bit 16 **PTIMEEN**: Timed mode enable
0: Disabled
1: Enabled

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **PATF0**: FF/00 pattern enable
Indicates whether the FF/00 pattern is enabled as output over the trace port
0: Disabled
1: Enabled

Bit 2 **PATA5**: AA/55 pattern is enable
Indicates whether the AA/55 pattern is enabled as output over the trace port
0: Disabled
1: Enabled

Bit 1 **PATW0**: Walking 0's pattern enable
Indicates whether the walking 0's pattern is enabled as output over the trace port
0: Disabled
1: Enabled

Bit 0 **PATW1**: Walking 1's pattern enable
Indicates whether the walking 1's pattern is enabled as output over the trace port
0: Disabled
1: Enabled

TPIU test pattern repeat counter register (TPIU_TPRCR)

Address offset: 0x208

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATTCOUNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PATTCOUNT[7:0]**: Number of TRACECLKIN cycles
The field provides a 8-bit counter value to indicate the number of TRACECLKIN cycles for which a pattern runs before it switches to the next pattern.

TPIU formatter and flush status register (TPIU_FFSR)

Address offset: 0x300

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCPRESENT	FTSTOPPED	FLINPROG
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TCPRESENT**: TRACECTL output pin availability

Indicates whether the optional TRACECTL output pin is available for use.

0: TRACECTL pin is not present in this device.

Bit 1 **FTSTOPPED**: Formatter stopped

The formatter has received a stop request signal and all trace data and post-amble is sent. Any additional trace data on the ATB interface is ignored.

0: Formatter has not stopped

1: Formatter has stopped

Bit 0 **FLINPROG**: Flush in progress

Indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the TPIU_FFCR register.

0: No flush in progress

1: Flush in progress

TPIU formatter and flush control register (TPIU_FFCR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	STOPTRIG	STOPFL	Res.	TRIGFL	TRIGEVT	TRIGIN	Res.	FONMAN	FONTRIG	FONFLIN	Res.	Res.	ENFCONT	ENFTC
		rw	rw		rw	rw	rw		rw	rw	rw			rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **STOPTRIG**: Stop on trigger event

0: No effect

1: Stop formatter when a trigger event occurs

Bit 12 **STOPFL**: Stop on flush

0: No effect

1: Stop formatter when flush is completed

Bit 11 Reserved, must be kept at reset value.

Bit 10 **TRIGFL**: Trigger on flush

0: No effect

1: Indicates a trigger in the trace stream when flush is completed

Bit 9 **TRIGEVT**: Trigger on trigger event

0: No effect

1: Indicates a trigger in the trace stream when trigger event occurs

Bit 8 **TRIGIN**: Trigger on trigger in

0: No effect

1: Indicates a trigger in the trace stream when the TRIGIN input from the system CTI is asserted.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FONMAN**: Generate a manual flush

0: No effect

1: Flush the trace

This bit is cleared automatically when the flush completes.

Bit 5 **FONTRIG**: Flush on trigger event

A trigger event occurs when the trigger counter reaches 0, or, if the trigger counter is 0, when the TRIGIN input from the system CTI is high.

0: No effect

1: Flush the trace if a trigger event occurs

Bit 4 **FONFLIN**: Flush on flush in

0: No effect

1: Flush the trace if the FLUSHIN input from the system CTI is asserted

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **ENFCONT:** Enable continuous formatting

0: Continuous formatting is disabled

1: Continuous formatting is enabled

Bit 0 **ENFTC**: Enable the embedding of triggers in formatted trace

0: Formatting is disabled

1: Formatting is enabled

TPIU formatter synchronization counter register (TPIU_FSCR)

Address offset: 0x400

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CYCCOUNT[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **CYCCOUNT[11:0]**: Enables effective use of TPAs

Enables effective use of different-sized TPAs without wasting large amounts of storage capacity of the capture device. This counter contains the number of formatter frames since the last synchronization packet of 128 bits. It is a 12-bit counter with a maximum count value of 4096. This equates to a synchronization every 65536 bytes, that is, 4096 packets x 16 bytes per packet. The default is set up for a synchronization packet every 1024 bytes, that is, every 64 formatter frames. If the formatter is configured for continuous mode, full and half-word sync frames are inserted during normal operation. Under these circumstances, the count value is the maximum number of complete frames between full synchronization packets.

TPIU claim tag set register (TPIU_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:
0000: No effect
XXX1: Set bit 0
XX1X: Set bit 1
X1XX: Set bit 2
1XXX: Set bit 3
Read:
0xF: Indicates there are four bits in claim tag

TPIU claim tag clear register (TPIU_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits

Write:
0000: No effect
XXX1: Clear bit 0
XX1X: Clear bit 1
X1XX: Clear bit 2
1XXX: Clear bit 3
Read: Returns current value of claim tag

TPIU lock access register (TPIU_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: TPIU register access enable

Enables write access to some TPIU registers by the processor cores (debuggers do not need to unlock the component)

0xC5ACCE55: Enable write access

Other values: Disable write access

TPIU lock status register (TPIU_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the TPIU_LAR register

0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock

This bit always returns zero when read by an external debugger.

0: Write access is permitted

1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Implementation of a lock control mechanism

The bit indicates whether a lock control mechanism is implemented. It always returns zero when read by an external debugger.

0: No lock control mechanism is available

1: Lock control mechanism is implemented

TPIU authentication status register (TPIU_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x0: Not implemented

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x0: Not implemented

TPIU device configuration register (TPIU_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 00A0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MAXNUM[4:0]				
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SWOUARTNRZ**: Support of SWO UART or NRZ
Indicates whether serial wire output, UART or NRZ, is supported.
0: Not supported

Bit 10 **SWOMAN**: Support of SWO Manchester format
Indicates whether serial wire output, Manchester encoded format, is supported.
0: Not supported

Bit 9 **TCLKDATA**: Support of trace clock plus data
0: Not supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of 2
0x2: FIFO size = 4 (16 bytes)

Bit 5 **CLKRELAT**: ATB clock and TRACECLKIN relation
Indicates the relationship between the ATB clock and TRACECLKIN (synchronous or asynchronous)
1: Asynchronous

Bits 4:0 **MAXNUM[4:0]**: Number/type of ATB input port multiplexing
0x0: None

TPIU device type identifier register (TPIU_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification
0x1: Trace port component

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x1: Trace sink component

TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x12: TPIU part number

TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: TPIU part number

TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 005B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x5: r1p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

TPIU CoreSight component identity register 0 (TPIU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

TPIU CoreSight component identity register 1 (TPIU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

TPIU CoreSight component identity register 2 (TPIU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

TPIU CoreSight component identity register 3 (TPIU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

52.6.15 TPIU register map and reset values

Table 623. TPIU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TPIU_SUPPSIZE	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x004	TPIU_CURPSIZE	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x008 to 0x0FC	Reserved																																
0x100	TPIU_SUPTRGM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGRUN	TRIGD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCOUNT8	Res.	Res.	Res.	MULT64K	MULT256	MULT16	MULT4	MULT2
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-	-	-	-	-	1	-	-	-	1	1	1	1	1
0x104	TPIU_TRGCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGCOUNT[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x108	TPIU_TRGMULT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MULT64K	MULT256	MULT16	MULT4	MULT2
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0
0x10C to 0x1FC	Reserved																																
0x200	TPIU_SUPTPM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCONTEN	PTIMEEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATF0	PATA5	PATW0	PATW1
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1
0x204	TPIU_CURTPM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCONTEN	PTIMEEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATF0	PATA5	PATW0	PATW1
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
0x208	TPIU_TPRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATTCOUNT[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x20C to 0x2FC	Reserved																																
0x300	TPIU_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCPRESENT	FTSTOPPED	FLINPROG
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	0

Table 623. TPIU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x304	TPIU_FFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOPTRIG	STOPFL	Res.	TRiGFL	TRiGEVt	TRIGIN	Res.	FONMAN	FONTRIG	FONFLIN	Res.	Res.	ENFCONT	ENFTC
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	-	0	0	0	-	0	0	0	-	-	0	0
0x308	TPIU_FSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CYCCOUNT[11:0]											
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	0	0	0	0	0
0x30C to 0xF9C	Reserved																																
0xFA0	TPIU_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1
0xFA4	TPIU_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
0xFB0	TPIU_LAR	ACCESS_W[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xFB4	TPIU_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	1
0xFB8	TPIU_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0xFBC to 0xFC4	Reserved																																
0xFC8	TPIU_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MUXNUM[4:0]				
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	0	1	0	0	0	0	0
0xFCC	TPIU_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYP[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	1	0	0	0	1
0xFD0	TPIU_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved																																

Table 623. TPIU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE0	TPIU_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	1	0	0	1	0
0xFE4	TPIU_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	1	0	0	1
0xFE8	TPIU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	0	1	1	0	1	1
0xFEC	TPIU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0xFF0	TPIU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	1	0	1
0xFF4	TPIU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	0	1	0	0	0	0
0xFF8	TPIU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	1
0xFFC	TPIU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	1

52.7 Serial wire output (SWO)

The SWO is a CoreSight component that formats the trace stream from the processor ITM and outputs it on the single wire TRACESWO.

Compared to the TPIU, the SWO contains:

- No formatter
- No pattern generator
- An 8-bit ATB input
- No synchronous trace output, that is, no TRACEDATA or TRACECLK pins
- No support for flush, because this is not required
- No support for triggering

The SWO output supports Manchester encoded and UART NRZ formats.

For more information about the serial wire output CoreSight™ component, refer to the Arm® CoreSight™ Components Technical Reference Manual *[4]*.

52.7.1 SWO registers

SWO current output divisor register (SWO_CODR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PRESCALER[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PRESCALER[12:0]**: SWO baud rate scaling

The baud rate is the trace clock frequency divided by (PRESCALER - 1). The baud rate changes instantly, so it is recommended to stop the trace source and wait until the port is idle before writing to this register.

SWO selected pin protocol register (SWO_SPPR)

Address offset: 0x0F0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPROT[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **PPROT[1:0]**: Pin protocol

0x0: Reserved

0x1: Manchester

0x2: NRZ

0x3: Reserved

SWO formatter and flush status register (SWO_FFSR)

Address offset: 0x300

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FTNONSTOP**: Change of settings without stopping formatter
1: Change of settings is allowed with formatter running

Bit 2 **TCPRESENT**: TRACECTL pin present on SWO
0: TRACECTL pin not present

Bit 1 **FTSTOPPED**: Formatter stopped
0: Formatter running
The bit always returns 0 as the SWO formatter cannot be stopped in this device.

Bit 0 **FLINPROG**: Flush in progress
0: Flush is not in progress
The bit always returns 0 as SWO flushing is not supported in this device.

SWO claim tag set register (SWO_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits
Write:
0000: No effect
XXX1: Set bit 0
XX1X: Set bit 1
X1XX: Set bit 2
1XXX: Set bit 3
Read:
0xF: Indicates there are four bits in claim tag

SWO claim tag clear register (SWO_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits

Write:
0000: No effect
XXX1: Clear bit 0
XX1X: Clear bit 1
X1XX: Clear bit 2
1XXX: Clear bit 3
Read: Returns current value of claim tag

SWO lock access register (SWO_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: SWO register write access enable

Enables write access to some SWO registers by the processor cores (debuggers do not need to unlock the component)
0xC5ACCE55: Enable write access
Other values: Disable write access

SWO lock status register (SWO_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the SWO_LAR register
0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock
This bit always returns zero when read by an external debugger.
0: Write access is permitted
1: Write access is blocked - only read access is permitted

Bit 0 **LOCKEXIST**: Implementation of a lock control mechanism
The bit indicates whether a lock control mechanism is implemented. It always returns zero when read by an external debugger.
0: No lock control mechanism available
1: Lock control mechanism is implemented

SWO authentication status register (SWO_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x0: Not implemented

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x0: Not implemented

SWO device configuration register (SWO_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 0EA0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MUXNUM[4:0]				
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SWOUARTNRZ**: SWO UART or NRZ support
Indicates whether serial wire output, UART or NRZ, is supported.
1: Supported

Bit 10 **SWOMAN**: SWO Manchester format support
Indicates whether serial wire output, Manchester encoded format, is supported.
1: Supported

Bit 9 **TCLKDATA**: Trace clock plus data support
Indicates whether trace clock plus data is supported
1: Supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of 2
0x2: FIFO size = 4 (16 bytes)

Bit 5 **CLKRELAT**: ATB clock to TRACECLKIN relation
Indicates the relationship between the ATB clock and TRACECLKIN (synchronous or asynchronous)
1: Asynchronous

Bits 4:0 **MUXNUM[4:0]**: Number/type of ATB input port multiplexing
0x0: None

SWO device type identifier register (SWO_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification
0x1: Trace port component

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x1: Trace sink component

SWO CoreSight peripheral identity register 4 (SWO_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

SWO CoreSight peripheral identity register 0 (SWO_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x14: SWO part number

SWO CoreSight peripheral identity register 1 (SWO_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: SWO part number

SWO CoreSight peripheral identity register 2 (SWO_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 002B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x2: r0p2

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

SWO CoreSight peripheral identity register 3 (SWO_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

SWO CoreSight component identity register 0 (SWO_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

SWO CoreSight component identity register 1 (SWO_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

SWO CoreSight component identity register 2 (SWO_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

SWO CoreSight component identity register 3 (SWO_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

52.7.2 SWO register map and reset values

Table 624. SWO register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	SWO_CODR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[12:0]												
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 624. SWO register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x014 to 0x0EC	Reserved	Reserved																															
0x0F0	SWO_SPPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPROT[1:0]	
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1
0x0F4 to 0x2FC	Reserved																																
0x300	SWO_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	0	0
0x304 to 0xF9C	Reserved																																
0xFA0	SWO_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1
0xFA4	SWO_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
0xFA8 to 0xFAC	Reserved																																
0xFB0	SWO_LAR	ACCESS_W[31;0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xFB4	SWO_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	1
0xFB8	SWO_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0xFBC to 0xFC4	Reserved																																
0xFC8	SWO_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MUXNUM[4:0]				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	1	0	0	0	0	0

Table 624. SWO register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	SWO_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0xFD0	SWO_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	SWO_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	1	0	1	0	0
0xFE4	SWO_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	1	0	0	1
0xFE8	SWO_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	1	0	1	0	1	1
0xFEC	SWO_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0xFF0	SWO_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	1	0	1
0xFF4	SWO_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	0	1	0	0	0	0
0xFF8	SWO_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	1
0xFFC	SWO_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	1

52.8 Trace replicator

The trace replicator is a CoreSight component that splits the ATB bus into two branches, so that the trace data can be routed to two different sinks. The replicator has two ATB master ports, and one ATB slave port. The trace packets received on the slave port are duplicated, and output on the master ports.

The replicator is used to route the trace coming from the ITM to the serial wire output (SWO) and to the trace funnel (CSTF). The slave port is connected to the ITM. Master port 0 is connected to the SWO. Master port 1 is connected to the trace funnel.

As data are received from the trace source, it is passed on to all trace sinks at the same time. By default, the replicator does not accept more data from the trace source until all the trace sinks have accepted this data. This has the impact of reducing the throughput of the replicator to match that of the slowest trace sink.

The trace packets output on each master port can be filtered according to the source ID, by programming REP_IDFILTERx registers. This may be required in the following examples:

- The ITM generates a high volume of trace packets, which are routed to the TPIU. The SWO is disabled. All packets on ITM replicator master port 0 must be disabled to avoid blocking the flow to master port 1.
- ITM trace packets are output to the SWO only, and the CSTF slave port connected to the ITM replicator is disabled. All packets on ITM replicator master port 1 must be disabled to avoid blocking the flow to master port 0.

For more information on the ATB replicator CoreSight component, refer to the Arm CoreSight SoC-400 Technical Reference Manual *[2]*.

52.8.1 Trace replicator registers

The register file base address for each replicator is defined by *Table 613: System ROM table*. The registers are the same for each replicator.

Replicator ID filter register 0 (REP_IDFILTER0)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID0_70_7F	ID0_60_6F	ID0_50_5F	ID0_40_4F	ID0_30_3F	ID0_20_2F	ID0_10_1F	ID0_0_F
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **ID0_70_7F**: Enable or disable ID filtering for IDs 0x70-0x7F
0: Transactions with these IDs are passed on to ATB master port 0.
1: Transactions with these IDs are discarded on master port 0.

Bit 6 **ID0_60_6F**: Enable or disable ID filtering for IDs 0x60-0x6F
0: Transactions with these IDs are passed on to ATB master port 0.
1: Transactions with these IDs are discarded on master port 0.

Bit 5 **ID0_50_5F**: Enable or disable ID filtering for IDs 0x50-0x5F
0: Transactions with these IDs are passed on to ATB master port 0.
1: Transactions with these IDs are discarded on master port 0.

Bit 4 **ID0_40_4F**: Enable or disable ID filtering for IDs 0x40-0x4F
0: Transactions with these IDs are passed on to ATB master port 0.
1: Transactions with these IDs are discarded on master port 0

Bit 3 **ID0_30_3F**: Enable or disable ID filtering for IDs 0x30-0x3F
0: Transactions with these IDs are passed on to ATB master port .0
1: Transactions with these IDs are discarded on master port 0.

Bit 2 **ID0_20_2F**: Enable or disable ID filtering for IDs 0x20-0x2F
0: Transactions with these IDs are passed on to ATB master port 0.
1: Transactions with these IDs are discarded on master port 0.

Bit 1 **ID0_10_1F**: Enable or disable ID filtering for IDs 0x10-0x1F
0: Transactions with these IDs are passed on to ATB master port 0.
1: Transactions with these IDs are discarded on master port 0.

Bit 0 **ID0_0_F**: Enable or disable ID filtering for IDs 0x0-0xF
0: Transactions with these IDs are passed on to ATB master port 0.
1: Transactions with these IDs are discarded on master port 0.

Replicator ID filter register 1 (REP_IDFILTER1)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID1_70_7F	ID1_60_6F	ID1_50_5F	ID1_40_4F	ID1_30_3F	ID1_20_2F	ID1_10_1F	ID1_0_F
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **ID1_70_7F**: Enable or disable ID filtering for IDs 0x70-0x7F
0: Transactions with these IDs are passed on to ATB master port 1.
1: Transactions with these IDs are discarded on master port 1.

Bit 6 **ID1_60_6F**: Enable or disable ID filtering for IDs 0x60-0x6F
0: Transactions with these IDs are passed on to ATB master port 1.
1: Transactions with these IDs are discarded on master port 1.

Bit 5 **ID1_50_5F**: Enable or disable ID filtering for IDs 0x50-0x5F
0: Transactions with these IDs are passed on to ATB master port 1.
1: Transactions with these IDs are discarded on master port 1.

Bit 4 **ID1_40_4F**: Enable or disable ID filtering for IDs 0x40-0x4F
0: Transactions with these IDs are passed on to ATB master port 1.
1: Transactions with these IDs are discarded on master port 1.

Bit 3 **ID1_30_3F**: Enable or disable ID filtering for IDs 0x30-0x3F
0: Transactions with these IDs are passed on to ATB master port 1.
1: Transactions with these IDs are discarded on master port 1.

Bit 2 **ID1_20_2F**: Enable or disable ID filtering for IDs 0x20-0x2F
0: Transactions with these IDs are passed on to ATB master port 1.
1: Transactions with these IDs are discarded on master port 1.

Bit 1 **ID1_10_1F**: Enable or disable ID filtering for IDs 0x10-0x1F
0: Transactions with these IDs are passed on to ATB master port 1.
1: Transactions with these IDs are discarded on master port 1.

Bit 0 **ID1_0_F**: Enable or disable ID filtering for IDs 0x0-0xF
0: Transactions with these IDs are passed on to ATB master port 1.
1: Transactions with these IDs are discarded on master port 1.

Replicator claim tag set register (REP_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits
Write:
0000: No effect
XXX1: Set bit 0.
XX1X: Set bit 1.
X1XX: Set bit 2.
1XXX: Set bit 3.
Read:
0xF: Indicates there are four bits in claim tag.

Replicator claim tag clear register (REP_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits
Write:
0000: No effect
XXX1: Clear bit 0.
XX1X: Clear bit 1.
X1XX: Clear bit 2.
1XXX: Clear bit 3.
Read: Returns current value of claim tag.

Replicator lock access register (REP_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: CSTF register write access enable
The field enables write access to some CSTF registers by the processor cores (debuggers do not need to unlock the component).
0xC5ACCE55: Enable write access.
Other values: Disable write access.

Replicator lock status register (REP_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of CSTF_LAR register
0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock
This bit always returns zero when read by an external debugger.
0: Write access is permitted.
1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.
0: No lock control mechanism exists.
1: Lock control mechanism is implemented.

Replicator authentication status register (REP_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x0: Not implemented

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x0: Not implemented

Replicator CoreSight device identity register (REP_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORTNUM[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PORTNUM[3:0]**: Number of master ports implemented
0x2: Two master ports

Replicator CoreSight device type identity register (REP_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0022

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVTYPEID[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DEVTYPEID[7:0]**: Device type identifier
0x22: Trace replicator

Replicator CoreSight peripheral identity register 4 (REP_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

Replicator CoreSight peripheral identity register 0 (REP_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0009

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x09: Replicator part number

Replicator CoreSight peripheral identity register 1 (REP_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: REP part number

Replicator CoreSight peripheral identity register 2 (REP_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 002B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x2: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm JEDEC code

Replicator CoreSight peripheral identity register 3 (REP_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

Replicator CoreSight component identity register 0 (REP_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

Replicator CoreSight component identity register 1 (REP_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

Replicator CoreSight component identity register 2 (REP_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

Replicator CoreSight component identity register 3 (REP_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

Trace replicator register map

Table 625. Replicator register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	REP_IDFILTER0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID0_70_7F	ID0_60_6F	ID0_50_5F	ID0_40_4F	ID0_30_3F	ID0_20_2F	ID0_10_1F	ID0_0_F
	Reset value																									0	0	0	0	0	0	0	0
0x004	REP_IDFILTER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID1_70_7F	ID1_60_6F	ID1_50_5F	ID1_40_4F	ID1_30_3F	ID1_20_2F	ID1_10_1F	ID1_0_F
	Reset value																									0	0	0	0	0	0	0	0
0x008-0xF9C	Reserved																																
0xFA0	REP_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	REP_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0xFA8-0xFAC	Reserved																																
0xFB0	REP_LAR	ACCESS_W[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0xFB4	REP_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	REP_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID [1:0]		SID [1:0]		NSNID [1:0]		NSID [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0xFC8	REP_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORTNUM [3:0]			
	Reset value																													0	0	1	0
0xFCC	REP_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVTYPEID[7:0]							
	Reset value																									0	0	1	0	0	0	1	0
0xFD0	REP_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved																																
0xFE0	REP_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	1	0	0	1
0xFE4	REP_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	REP_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	1	0	1	0	1	1

Table 625. Replicator register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFEC	REP_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	REP_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	REP_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	REP_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	REP_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

52.9 Microcontroller debug unit (DBGMCU)

The DBGMCU component contains a number of registers that control the power and clock behavior in Debug mode. Specifically it allows the debugger, or debug software, to:

- Maintain the clock and power to the processor cores when in low-power modes (sleep, stop or standby)
- Maintain the clock and power to the system debug and trace components when in low low-power modes
- Stop the clock to certain peripherals (SMBUS timeout, Watchdogs, Timers, RTC) when the processor core is stopped in Debug mode. For timers having complementary outputs, these outputs are disabled (as if the MOE bit was reset) for safety purposes when the counter is stopped (TIM1/8/15/16/17/20 = 1 in DBGMCU_APB2FZR and DBGMCU_APB1FZR).

The DBGMCU registers are not reset by a system reset, only by a power on reset. They are accessible to the debugger via the APB-D bus at base address 0xE00E1000. They are also accessible by the processor core at base address 0x5C001000.

52.9.1 DBGMCU registers

DBGMCU identity code register (DBGMCU_IDCODE)

Address offset: 0x000

Reset value: 0x1001 647B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID[15:0]**: 0x1001 = revision Z

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]**: Device ID
0x47B: STM32H7Pxxx

DBGMCU configuration register (DBGMCU_CR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TRGOEN	Res.	Res.	Res.	Res.	Res.	Res.	DBGCKEN	TRACECLKEN	Res.	Res.	Res.	DCRT
			rw							rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_STANDBY	DBG_STOP	DBG_SLEEP
													rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **TRGOEN**: External trigger output enable

This bit controls the direction of the bi-directional trigger pin, TRGIO.

0: Input - TRGIO is connected to TRGIN

1: Output - TRGIO is connected to TRGOUT

Bits 27:22 Reserved, must be kept at reset value.

Bit 21 **DBGCKEN**: Debug clock enable

This bit allows clocks of the debug components (excluding those in the processor core) to be switched off if they are not needed.

0: Disabled - debug components are disabled and their clocks gated

1: Enabled - debug components are clocked whenever the corresponding domain clock (CK_HCLK) is active

Bit 20 **TRACECLKEN**: Trace port clock enable.

This bit enables the trace port clock, traceclkin.

0: Disabled - traceclkin is disabled

1: Enabled - traceclkin is active

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **DCRT**: Debug credentials reset type

This bit selects which type of reset is used to revoke the debug authentication credentials

0: System reset

1: Power reset

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY**: Debug in Standby mode enable

0: Normal operation - all clocks are disabled and the core powered down automatically in Standby mode.

1: Automatic clock stop/power-down disabled - all active clocks and oscillators continue to run during Standby mode, and the core supply is maintained, allowing full debug capability. On exit from Standby mode, a core reset is performed.

Bit 1 **DBG_STOP**: Debug in Stop mode enable

0: Normal operation - all clocks are disabled automatically in Stop mode

1: Automatic clock stop disabled - all active clocks and oscillators continue to run during Stop mode, allowing full debug capability. On exit from Stop mode, the clock settings are set to the Stop mode exit state.

Bit 0 **DBG_SLEEP**: Debug in Sleep mode enable

0: Normal operation - the processor clock is stopped automatically in Sleep mode

1: Automatic clock stop disabled - processor clock continues to run, allowing full debug capability

DBGMCU AHB5 peripheral freeze register (DBGMCU_AHB5FZR)

Address offset: 0x01C

Reset value: 0x0000 0000

Description: This register is accessible to the debugger after successful authentication. Prior to this, debugger accesses are ignored. It is always accessible to software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPU	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HPDMA_15	HPDMA_14	HPDMA_13	HPDMA_12	HPDMA_11	HPDMA_10	HPDMA_9	HPDMA_8	HPDMA_7	HPDMA_6	HPDMA_5	HPDMA_4	HPDMA_3	HPDMA_2	HPDMA_1	HPDMA_0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **NPU**: NPU stop in debug

0: Normal operation. NPU continues to operate while the CPU is in debug mode.

1: Stop in debug. NPU is frozen while the CPU is in debug mode.

Bits 30:16 Reserved, must be kept at reset value.

Bit 15 **HPDMA_15**: HPDMA channel 15 stop in debug

0: Normal operation. HPDMA channel 15 continues to operate while the CPU is in debug mode.

1: Stop in debug. HPDMA channel 15 is frozen while the CPU is in debug mode.

Bit 14 **HPDMA_14**: HPDMA channel 14 stop in debug

0: Normal operation. HPDMA channel 14 continues to operate while the CPU is in debug mode.

1: Stop in debug. HPDMA channel 14 is frozen while the CPU s in debug mode.

Bit 13 **HPDMA_13**: HPDMA channel 13 stop in debug

0: Normal operation. HPDMA channel 13 continues to operate while the CPU is in debug mode.

1: Stop in debug. HPDMA channel 13 is frozen while the CPU is in debug mode.

Bit 12 **HPDMA_12**: HPDMA channel 12 stop in debug

0: Normal operation. HPDMA channel 12 continues to operate while the CPU is in debug mode.

1: Stop in debug. HPDMA channel 12 is frozen while the CPU is in debug mode.

Bit 11 **HPDMA_11**: HPDMA channel 11 stop in debug
0: Normal operation. HPDMA channel 11 continues to operate while the CPU is in debug mode.
1: Stop in debug. HPDMA channel 11 is frozen while the CPU is in debug mode.

Bit 10 **HPDMA_10**: HPDMA channel 10 stop in debug
0: Normal operation. HPDMA channel 10 continues to operate while the CPU is in debug mode.
1: Stop in debug. HPDMA channel 10 is frozen while the CPU is in debug mode.

Bit 9 **HPDMA_9**: HPDMA channel 9 stop in debug
0: Normal operation. HPDMA channel 9 continues to operate while the CPU is in debug mode.
1: Stop in debug. HPDMA channel 9 is frozen while the CPU is in debug mode.

Bit 8 **HPDMA_8**: HPDMA channel 8 stop in debug
0: Normal operation. HPDMA channel 8 continues to operate while the CPU is in debug mode.
1: Stop in debug. HPDMA channel _ is frozen while the CPU is in debug mode.

Bit 7 **HPDMA_7**: HPDMA channel 7 stop in debug
0: Normal operation. HPDMA channel 7 continues to operate while the CPU is in debug mode.
1: Stop in debug. HPDMA channel 7 is frozen while the CPU is in debug mode.

Bit 6 **HPDMA_6**: HPDMA channel 6 stop in debug
0: Normal operation. HPDMA channel 6 continues to operate while the CPU is in debug mode.
1: Stop in debug. HPDMA channel 6 is frozen while the CPU is in debug mode.

Bit 5 **HPDMA_5**: HPDMA channel 5 stop in debug
0: Normal operation. HPDMA channel 5 continues to operate while the CPU is in debug mode.
1: Stop in debug. HPDMA channel 5 is frozen while the CPU is in debug mode.

Bit 4 **HPDMA_4**: HPDMA channel 4 stop in debug
0: Normal operation. HPDMA channel 4 continues to operate while the CPU is in debug mode.
1: Stop in debug. HPDMA channel 4 is frozen while the CPU is in debug mode.

Bit 3 **HPDMA_3**: HPDMA channel 3 stop in debug
0: Normal operation. HPDMA channel 3 continues to operate while the CPU is in debug mode.
1: Stop in debug. HPDMA channel 3 is frozen while the CPU is in debug mode.

Bit 2 **HPDMA_2**: HPDMA channel 2 stop in debug
0: Normal operation. HPDMA channel 2 continues to operate while the CPU is in debug mode.
1: Stop in debug. HPDMA channel 2 is frozen while the CPU is in debug mode.

Bit 1 **HPDMA_1**: HPDMA channel 1 stop in debug
0: Normal operation. HPDMA channel 1continues to operate while the CPU is in debug mode.
1: Stop in debug. HPDMA channel 1 is frozen while the CPU is in debug mode.

Bit 0 **HPDMA_0**: HPDMA channel 0 stop in debug
0: Normal operation. HPDMA channel 0 continues to operate while the CPU is in debug mode.
1: Stop in debug. HPDMA channel 0 is frozen while the CPU is in debug mode.

DBGMCU AHB1 peripheral freeze register (DBGMCU_AHB1FZR)

Address offset: 0x024

Reset value: 0x0000 0000

Description: This register is accessible to the debugger after successful authentication. Prior to this, debugger accesses are ignored. It is always accessible to software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPDMA_15	GPDMA_14	GPDMA_13	GPDMA_12	GPDMA_11	GPDMA_10	GPDMA_9	GPDMA_8	GPDMA_7	GPDMA_6	GPDMA_5	GPDMA_4	GPDMA_3	GPDMA_2	GPDMA_1	GPDMA_0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **GPDMA_15**: GPDMA channel 15 stop in debug
0: Normal operation. GPDMA channel 15 continues to operate while the CPU is in debug mode.
1: Stop in debug. GPDMA channel 15 is frozen while the CPU is in debug mode.

Bit 14 **GPDMA_14**: GPDMA channel 14 stop in debug
0: Normal operation. GPDMA channel 14 continues to operate while the CPU is in debug mode.
1: Stop in debug. GPDMA channel 14 is frozen while the CPU s in debug mode.

Bit 13 **GPDMA_13**: GPDMA channel 13 stop in debug
0: Normal operation. GPDMA channel 13 continues to operate while the CPU is in debug mode.
1: Stop in debug. GPDMA channel 13 is frozen while the CPU is in debug mode.

Bit 12 **GPDMA_12**: GPDMA channel 12 stop in debug
0: Normal operation. GPDMA channel 12 continues to operate while the CPU is in debug mode.
1: Stop in debug. GPDMA channel 12 is frozen while the CPU is in debug mode.

Bit 11 **GPDMA_11**: GPDMA channel 11 stop in debug
0: Normal operation. GPDMA channel 11 continues to operate while the CPU is in debug mode.
1: Stop in debug. GPDMA channel 11 is frozen while the CPU is in debug mode.

Bit 10 **GPDMA_10**: GPDMA channel 10 stop in debug
0: Normal operation. GPDMA channel 10 continues to operate while the CPU is in debug mode.
1: Stop in debug. GPDMA channel 10 is frozen while the CPU is in debug mode.

Bit 9 **GPDMA_9**: GPDMA channel 9 stop in debug
0: Normal operation. GPDMA channel 9 continues to operate while the CPU is in debug mode.
1: Stop in debug. GPDMA channel 9 is frozen while the CPU is in debug mode.

Bit 8 **GPDMA_8**: GPDMA channel 8 stop in debug
0: Normal operation. GPDMA channel 8 continues to operate while the CPU is in debug mode.
1: Stop in debug. GPDMA channel _ is frozen while the CPU is in debug mode.

Bit 7 **GPDMA_7**: GPDMA channel 7 stop in debug
0: Normal operation. GPDMA channel 7 continues to operate while the CPU is in debug mode.
1: Stop in debug. GPDMA channel 7 is frozen while the CPU is in debug mode.

Bit 6 **GPDMA_6**: GPDMA channel 6 stop in debug
0: Normal operation. GPDMA channel 6 continues to operate while the CPU is in debug mode.
1: Stop in debug. GPDMA channel 6 is frozen while the CPU is in debug mode.

Bit 5 **GPDMA_5**: GPDMA channel 5 stop in debug
0: Normal operation. GPDMA channel 5 continues to operate while the CPU is in debug mode.
1: Stop in debug. GPDMA channel 5 is frozen while the CPU is in debug mode.

Bit 4 **GPDMA_4**: GPDMA channel 4 stop in debug
0: Normal operation. GPDMA channel 4 continues to operate while the CPU is in debug mode.
1: Stop in debug. GPDMA channel 4 is frozen while the CPU is in debug mode.

Bit 3 **GPDMA_3**: GPDMA channel 3 stop in debug
0: Normal operation. GPDMA channel 3 continues to operate while the CPU is in debug mode.
1: Stop in debug. GPDMA channel 3 is frozen while the CPU is in debug mode.

Bit 2 **GPDMA_2**: GPDMA channel 2 stop in debug
0: Normal operation. GPDMA channel 2 continues to operate while the CPU is in debug mode.
1: Stop in debug. GPDMA channel 2 is frozen while the CPU is in debug mode.

Bit 1 **GPDMA_1**: GPDMA channel 1 stop in debug
0: Normal operation. GPDMA channel 1continues to operate while the CPU is in debug mode.
1: Stop in debug. GPDMA channel 1 is frozen while the CPU is in debug mode.

Bit 0 **GPDMA_0**: GPDMA channel 0 stop in debug
0: Normal operation. GPDMA channel 0 continues to operate while the CPU is in debug mode.
1: Stop in debug. GPDMA channel 0 is frozen while the CPU is in debug mode.

DBGMCU APB1 peripheral freeze register (DBGMCU_APB1FZR)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	I3C1	I2C3	I2C2	I2C1	Res.	Res.	Res.	Res.	Res.
							rw	rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	TIM11	TIM10	WWDG	Res.	LPTIM1	TIM14	TIM13	TIM12	TIM7	TIM6	TIM5	TIM4	TIM3	TIM2
		rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **I3C1**: I3C1 SMBUS timeout stop in debug
0: Normal operation - I3C1 SMBUS timeout continues operating while the core is in Debug mode
1: Stop in debug - I3C1 SMBUS timeout is frozen while Cortex-M7 is in Debug mode

Bit 23 **I2C3**: I2C3 SMBUS timeout stop in debug
0: Normal operation - I2C3 SMBUS timeout continues operating while the core is in Debug mode
1: Stop in debug - I2C3 SMBUS timeout is frozen while Cortex-M7 is in Debug mode

Bit 22 **I2C2**: I2C2 SMBUS timeout stop in debug
0: Normal operation - I2C2 SMBUS timeout continues operating while the core is in Debug mode
1: Stop in debug - I2C2 SMBUS timeout is frozen while Cortex-M7 is in Debug mode

Bit 21 **I2C1**: I2C1 SMBUS timeout stop in debug
0: Normal operation - I2C1 SMBUS timeout continues operating while the core is in Debug mode
1: Stop in debug - I2C1 SMBUS timeout is frozen while the core is in Debug mode

Bits 20:14 Reserved, must be kept at reset value.

Bit 13 **TIM11**: TIM11 stop in debug
0: Normal operation - TIM11 continues operating while the core is in Debug mode
1: Stop in debug - TIM11 is frozen while Cortex-M7 is in Debug mode

Bit 12 **TIM10**: TIM10 stop in debug
0: Normal operation - TIM10 continues operating while the core is in Debug mode
1: Stop in debug - TIM10 is frozen while Cortex-M7 is in Debug mode

Bit 11 **WWDG**: WWDG stop in debug
0: Normal operation - WWDG continues to operate while the Cortex-M7 is in Debug mode
1: Stop in debug - WWDG is frozen while the Cortex-M7 is in Debug mode

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1**: LPTIM1 stop in debug
0: Normal operation - LPTIM1 continues operating while the core is in Debug mode
1: Stop in debug - LPTIM1 is frozen while Cortex-M7 is in Debug mode

Bit 8 **TIM14**: TIM14 stop in debug
0: Normal operation - TIM14 continues operating while the core is in Debug mode
1: Stop in debug - TIM14 is frozen while Cortex-M7 is in Debug mode

Bit 7 **TIM13**: TIM13 stop in debug
0: Normal operation - TIM13 continues operating while the core is in Debug mode
1: Stop in debug - TIM13 is frozen while Cortex-M7 is in Debug mode

Bit 6 **TIM12**: TIM12 stop in debug
0: Normal operation - TIM12 continues operatinge while the core is in Debug mode
1: Stop in debug - TIM12 is frozen while Cortex-M7 is in Debug mode

Bit 5 **TIM7**: TIM7 stop in debug
0: Normal operation - TIM7 continues operatinge while the core is in Debug mode
1: Stop in debug - TIM7 is frozen while Cortex-M7 is in Debug mode

Bit 4 **TIM6**: TIM6 stop in debug
0: Normal operation - TIM6 continues operating while the core is in Debug mode
1: Stop in debug - TIM6 is frozen while Cortex-M7 is in Debug mode

Bit 3 **TIM5**: TIM5 stop in debug
0: Normal operation - TIM5 continues operating while the core is in Debug mode
1: Stop in debug - TIM5 is frozen while Cortex-M7 is in Debug mode

Bit 2 **TIM4**: TIM4 stop in debug

0: Normal operation - TIM4 continues operating while the core is in Debug mode

1: Stop in debug - TIM4 is frozen while Cortex-M7 is in Debug mode

Bit 1 **TIM3**: TIM3 stop in debug

0: Normal operation - TIM3 continues operating while the core is in Debug mode

1: Stop in debug - TIM3 is frozen while Cortex-M7 is in Debug mode

Bit 0 **TIM2**: TIM2 stop in debug

0: Normal operation - TIM2 continues operating while the core is in Debug mode

1: Stop in debug - TIM2 is frozen while Cortex-M7 is in Debug mode

DBGMCU APB2 peripheral freeze register (DBGMCU_APB2FZ)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	HRTIM2	Res.	Res.	Res.	HRTIM1	Res.	Res.	Res.	Res.	TIM9	TIM17	TIM16	TIM15
			rw				rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM20	TIM8	TIM1
													rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **HRTIM2**: HRTIM2 stop in debug

0: Normal operation - HRTIM2 continues operating while the core is in Debug mode

1: Stop in debug - HRTIM2 is frozen and HRTIM2 outputs are held in a “safe” state while Cortex-M7 is in Debug mode

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **HRTIM1**: HRTIM1 stop in debug

0: Normal operation - HRTIM1 continues operating while the core is in Debug mode

1: Stop in debug - HRTIM1 is frozen and HRTIM1 outputs are held in a “safe” state while Cortex-M7 is in Debug mode

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **TIM9**: TIM9 stop in debug

0: Normal operation - TIM9 continues operating while the core is in Debug mode

1: Stop in debug - TIM9 is frozen and TIM9 outputs are disabled while Cortex-M7 is in Debug mode

Bit 18 **TIM17**: TIM17 stop in debug

0: Normal operation - TIM17 continues operating while the core is in Debug mode

1: Stop in debug - TIM17 is frozen and TIM17 outputs are disabled while Cortex-M7 is in Debug mode

Bit 17 **TIM16**: TIM16 stop in debug

0: Normal operation - TIM16 continues operating while the core is in Debug mode

1: Stop in debug - TIM16 is frozen and TIM16 outputs are disabled while Cortex-M7 is in Debug mode

Bit 16 **TIM15**: TIM15 stop in debug
- 0: Normal operation - TIM15 continues operating while the core is in Debug mode
- 1: Stop in debug - TIM15 is frozen and TIM15 outputs are disabled while Cortex-M7 is in Debug mode

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **TIM20**: TIM20/PWM20 stop in debug
- 0: Normal operation - TIM20 continues operating while the core is in Debug mode
- 1: Stop in debug - TIM20 is frozen and TIM20 outputs are disabled while Cortex-M7 is in Debug mode.

Bit 1 **TIM8**: TIM8/PWM8 stop in debug
- 0: Normal operation - TIM8 continues operating while the core is in Debug mode
- 1: Stop in debug - TIM8 is frozen and TIM8 outputs are disabled while Cortex-M7 is in Debug mode.

Bit 0 **TIM1**: TIM1/PWM1 stop in debug
- 0: Normal operation - TIM1 continues operating while the core is in Debug mode
- 1: Stop in debug - TIM1 is frozen and TIM1 outputs are disabled while Cortex-M7 is in Debug mode.

DBGMCU APB4 peripheral freeze register (DBGMCU_APB4FZR)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG	Res.	RTC
													rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LPTIM5	LPTIM4	LPTIM3	LPTIM2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw	rw	rw	rw									

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **IWDG**: Independent watchdog for stop in debug
- 0: Normal operation - watchdog continues counting while the core is in Debug mode
- 1: Stop in debug - watchdog is frozen while Cortex-M7 is in Debug mode

Bit 17 Reserved, must be kept at reset value.

Bit 16 **RTC**: RTC stop in debug
- 0: Normal operation - RTC continues operating while the core is in Debug mode
- 1: Stop in debug - RTC is frozen while Cortex-M7 is in Debug mode

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5**: LPTIM5 stop in debug
- 0: Normal operation - LPTIM5 continues operating while the core is in Debug mode
- 1: Stop in debug - LPTIM5 is frozen while Cortex-M7 is in Debug mode

Bit 11 **LPTIM4**: LPTIM4 stop in debug
- 0: Normal operation - LPTIM4 continues operating while the core is in Debug mode
- 1: Stop in debug - LPTIM4 is frozen while Cortex-M7 is in Debug mode

Bit 10 **LPTIM3**: LPTIM3 stop in debug
0: Normal operation - LPTIM2 continues operating while the core is in Debug mode
1: Stop in debug - LPTIM2 is frozen while Cortex-M7 is in Debug mode

Bit 9 **LPTIM2**: LPTIM2 stop in debug
0: Normal operation - LPTIM2 continues operating while the core is in Debug mode
1: Stop in debug - LPTIM2 is frozen while Cortex-M7 is in Debug mode

Bits 8:0 Reserved, must be kept at reset value.

DBGMCU EDBGRQ input control (DBGMCU_EDBGRQ)

Address offset: 0x080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_CPU
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DBG_CPU**: Drives the CPU external debug request (EDBGRQ) signal. This signal can be used to halt the CPU in debug on a system reset, instead of writing to the DEMCR.VC_CORERESET bit in the CPU
0: External debug request de-asserted
1: The external debug request, EDBGRQ, is asserted when the system reset is active and remains asserted until the cpu is Halted.

DBGMCU status register (DBGMCU_SR)

Address offset: 0x0FC

Reset value: 0x0001 0003

Description: This register is always accessible

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AP_ENABLED[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AP_PRESENT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **AP_ENABLED[15:0]**: Bit n identifies whether access port AP n is open (can be accessed via the debug port) or locked (debug access to the AP is blocked)
Bit n = 0: APn locked
Bit n = 1: APn enabled

Bits 15:0 **AP_PRESENT[15:0]**: Bit n identifies whether access port AP n is present in the device
Bit n = 0: APn absent
Bit n = 1: APn present

DBGMCU debug authentication mailbox host register (DBGMCU_DBG_AUTH_HOST)

Address offset: 0x100

Reset value: 0xXXXX XXXX

Description: This register is read only when accessed by the CPU, writes have no effect. It register can be written and read by an external debugger.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MESSAGE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MESSAGE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MESSAGE[31:0]**: Debug host to device mailbox message.
During debug authentication the debug host communicates with the device via this register.

DBGMCU debug authentication mailbox device register (DBGMCU_DBG_AUTH_DEVICE)

Address offset: 0x104

Reset value: 0xXXXX XXXX

Description: This register is read only when accessed via the debug port, writes have no effect. This register can be written and read by the CPU.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MESSAGE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MESSAGE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MESSAGE[31:0]**: Device to debug host mailbox message.
During debug authentication the device communicates with the debug host via this register.

DBGMCU debug authentication mailbox acknowledge register (DBGMCU_DBG_AUTH_ACK)

Address offset: 0x108

Reset value: 0x0000 0000

Description: This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEV_ACK	HOST_ACK
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DEV_ACK**: Device to device acknowledge

The host sets this bit to indicate that it has placed a message in the DBGMCU_DBG_AUTH_HOST register. It is reset by the device after reading the message

Bit 0 **HOST_ACK**: Host to device acknowledge

The device sets this bit to indicate that it has placed a message in the DBGMCU_DBG_AUTH_DEVICE register. It should be reset by the host after reading the message

DBGMCU peripheral identity register 4 (DBGMCU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size

0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code

0x0: STMicroelectronics JEDEC code

DBGMCU peripheral identity register 0 (DBGMCU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x0: DBGMCU

DBGMCU peripheral identity register 1 (DBGMCU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x0: DBGMCU

DBGMCU peripheral identity register 2 (DBGMCU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: Rev 0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x2: STMicroelectronics JEDEC code

DBGMCU peripheral identity register 3 (DBGMCU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

DBGMCU component identity register (DBGMCU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

DBGMCU component identity register (DBGMCU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0xF: System component with non-standard register layout

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

DBGMCU component identity register (DBGMCU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

DBGMCU component identity register (DBGMCU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]:** Component ID field, bits [31:24]
0xB1: Common ID value

52.9.2 DBGMCU register map and reset values

Table 626. DBGMCU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DBGMCU_IDCODE	REV_ID[15:0]																Res.	Res.	Res.	Res.	DEV_ID[11:0]											
	Reset value	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0					0	1	0	0	0	1	1	1	1	0	1	1
0x004	DBGMCU_CR	Res.	Res.	Res.	TRGOEN	Res.	Res.	Res.	Res.	Res.	Res.	DBGCKEN	TRACECLKEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_STANDBY	DBG_STOP	DBG_SLEEP
	Reset value				0							0	0																		1	0	0
0x008 to 0x018	Reserved																																
0x01C	DBGMCU_AHB5FZR	NPU	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HPDMA_15	HPDMA_14	HPDMA_13	HPDMA_12	HPDMA_11	HPDMA_10	HPDMA_9	HPDMA_8	HPDMA_7	HPDMA_6	HPDMA_5	HPDMA_4	HPDMA_3	HPDMA_2	HPDMA_1	HPDMA_0
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	Reserved																																
0x024	DBGMCU_AHB1FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA_15	GPDMA_14	GPDMA_13	GPDMA_12	GPDMA_11	GPDMA_10	GPDMA_9	GPDMA_8	GPDMA_7	GPDMA_6	GPDMA_5	GPDMA_4	GPDMA_3	GPDMA_2	GPDMA_1	GPDMA_0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28 to 0x038	Reserved																																
0x03C	DBGMCU_APB1FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I3C1	I2C3	I2C2	I2C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM11	TIM10	WWDG1	Res.	LPTIM1	TIM14	TIM13	TIM12	TIM7	TIM6	TIM5	TIM4	TIM3	TIM2
	Reset value								0	0	0	0										0		0	0	0	0	0	0	0	0	0	0
0x040 to 0x48	Reserved																																
0x044	DBGMCU_APB1HFZ1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM24	TIM23	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0	0																							
0x04C	DBGMCU_APB2FZ1	Res.	Res.	Res.	HRTIM2	Res.	Res.	HRTIM1	Res.	Res.	Res.	Res.	Res.	TIM19	TIM17	TIM16	TIM15	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM20	TIM8	TIM1
	Reset value				0			0						0	0	0	0														0	0	0
0x050	Reserved																																
0x054	DBGMCU_APB4FZ1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG1	Res.	RTC	Res.	Res.	Res.	LPTIM5	LPTIM4	LPTIM3	LPTIM2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0		0				0	0	0	0									
0x058 to 0x7C	Reserved																																

Table 626. DBGMCU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x080	DBGMCU_EDBGRQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_CPU
	Reset value																																0
0x084 to 0xF8	Reserved																																
0xFC	DBGMCU_SR	AP_ENABLED[15:0]																AP_PRESENT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0x100	DBGMCU_AUTH_HOST	MESSAGE[31:0]																															
		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x104	DBGMCU_AUTH_DEV	MESSAGE[31:0]																															
		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x108	DBGMCU_AUTH_ACK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEV_ACK	HOIST_ACK
																																0	0
0x10C to 0xFCC	Reserved																																
0xFD0	DBGMCU_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFE0	DBGMCU_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0xFE4	DBGMCU_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFE8	DBGMCU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
0xFEC	DBGMCU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 626. DBGMCU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF0	DBGMCU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
0xFF4	DBGMCU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0xFF8	DBGMCU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
0xFFC	DBGMCU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1

52.10 Cortex-M7 debug functional description

The Cortex-M7 subsystem features the following CoreSight™ components:

- ROM tables
- System control space (SCS)
- Breakpoint unit (FPB)
- Data watchpoint and trace unit (DWT)
- Instrumentation trace macrocell (ITM)
- Embedded trace macrocell (ETM)
- Cross trigger interface (CTI)

These components are accessible by the debugger via the Cortex-M7 AHB-AP and its associated AHBD bus.

52.10.1 Cortex-M7 ROM tables

The ROM table is a CoreSight™ component that contains the base addresses of all the CoreSight debug components accessible via the AHBD. These tables allow a debugger to discover the topology of the CoreSight system automatically.

There are two ROM tables in the Cortex-M7 sub-system:

- Cortex-M7 CPU ROM table

  This table is pointed to by the BASE register in the Cortex-M7 AHB-AP. It contains the base address pointers for the ETM and CTI, as well as for the Cortex-M7 PPB ROM table.
- Cortex-M7 PPB (private peripheral bus) ROM table

  This table contains pointers to the Cortex-M7 System Control Space registers allowing the debugger to identify the CPU core, as well as to the remaining CoreSight components in the Cortex-M7 subsystem: FPB, DWT and ITM.

The CPU ROM table occupies a 4-Kbyte, 32-bit wide chunk of AHBD address space, from 0xE00FE000 to 0xE00FEFFC.

Table 627. Cortex-M7 CPU ROM table

Address in ROM table	Component name	Component base address	Component address offset	Size	Entry
0xE00FE000	Cortex-M7 PPB ROM table	0xE00FF000	0x00001000	4 Kbyte	0x00001003
0xE00FE004	Cortex-M7 ETM	0xE0041000	0xFFF43000	4 Kbyte	0xFFF43003
0xE00FE008	Cortex-M7 CTI	0xE0043000	0xFFF45000	4 Kbyte	0xFFF45003
0xE00FE00C	Reserved	-	-	-	0x1FF02002
0xE00FE010	Top of table	-	-	-	0x00000000
0xE00FE010 to 0xE00FEFC8	Reserved	-	-	-	0x00000000
0xE00FEFCC to 0xE00FEFFC	ROM table registers	-	-	-	See *Table 629*

The Cortex-M7 PPB ROM table occupies a 4-Kbyte, 32-bit wide chunk of APB-D address space, from 0xE00FF000 to 0xE00FFFFC.

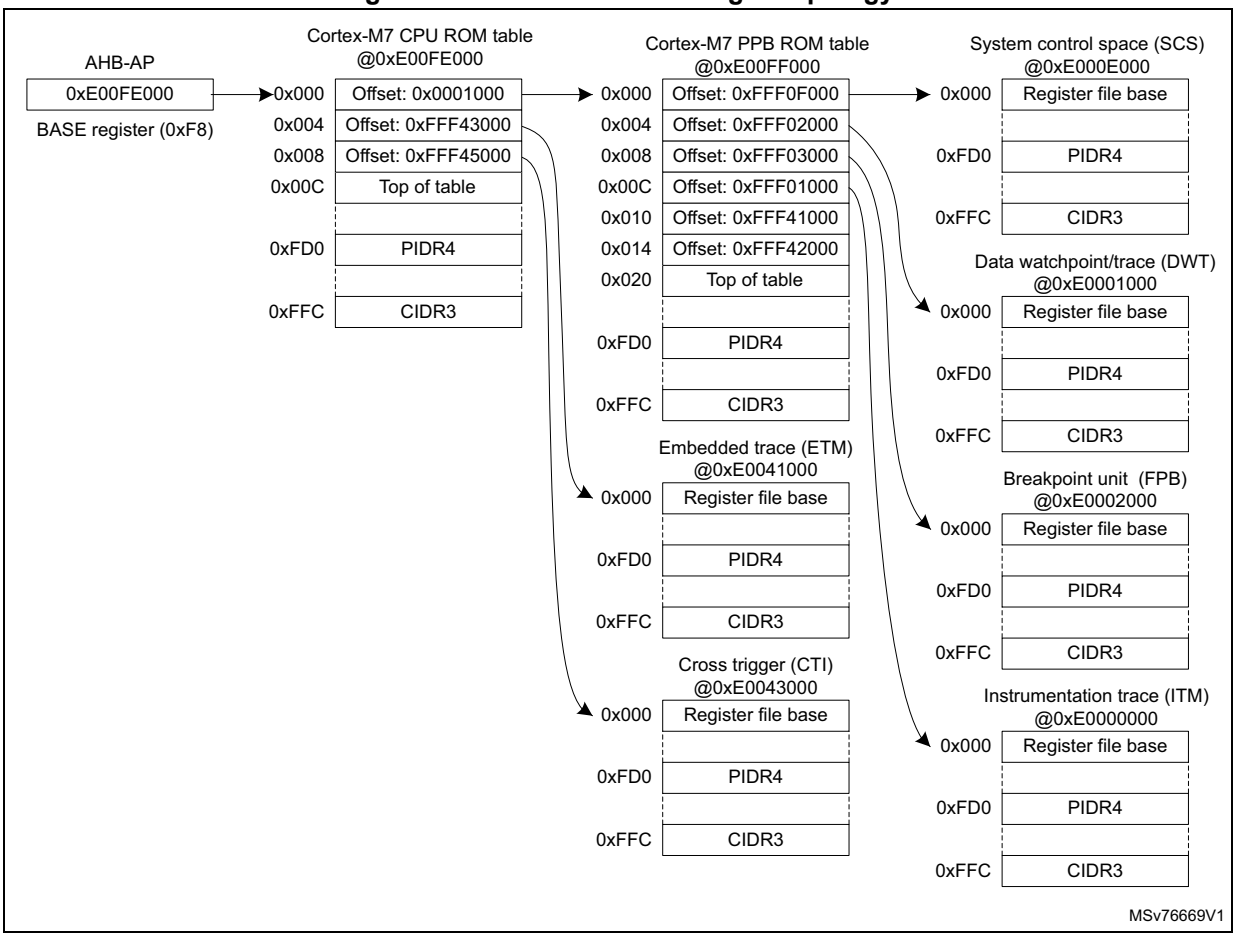
Table 628. Cortex-M7 PPB ROM table

Address in ROM table	Component name	Component base address	Component address offset	Size	Entry
0xE00FF000	SCS	0xE000E000	0xFFF0F000	4 Kbyte	0xFFF0F003
0xE00FF004	DWT	0xE0001000	0xFFF02000	4 Kbyte	0xFFF02003
0xE00FF008	FPB	0xE0002000	0xFFF03000	4 Kbyte	0xFFF03003
0xE00FF00C	ITM	0xE0000000	0xFFF01000	4 Kbyte	0xFFF01003
0xE00FF010	TPIU(1)	0xE0040000	0xFFF41000	4 Kbyte	0xFFF41002
0xE00FF014	ETM(1)	0xE0041000	0xFFF42000	4 Kbyte	0xFFF42002
0xE00FF018	Top of table	-	-	-	0x00000000
0xE00FF01C to 0xE00FFFC8	Reserved	-	-	-	0x00000000
0xE00FFFCC to 0xE00FFFFC	ROM table registers	-	-	-	See *Table 630*

1. The TPIU and ETM are included in this table by default, but bit 0 is reset to indicate that they are not present.

The topology for the CoreSight™ components in the Cortex-M7 subsystem is shown in *Figure 1051*.

Figure 1051. Cortex-M7 CoreSight Topology



52.10.2 Cortex-M7 CPU ROM registers

CPU ROM memory type register (M7_CPUROM_MEMTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory presence
1: System memory is present on this bus

CPU ROM CoreSight peripheral identity register 4 (M7_CPUROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC continuation code

CPU ROM CoreSight peripheral identity register 0 (M7_CPUROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0085

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x85: STM32H7Pxxx

CPU ROM CoreSight peripheral identity register 1 (M7_CPUROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0x0: STMicroelectronics

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x4: STM32H7Pxxx

CPU ROM CoreSight peripheral identity register 2 (M7_CPUROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: Rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x2: STMicroelectronics

CPU ROM CoreSight peripheral identity register 3 (M7_CPUROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

CPU ROM CoreSight component identity register 0 (M7_CPUROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

CPU ROM CoreSight component identity register 1 (M7_CPUROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

CPU ROM CoreSight component identity register 2 (M7_CPUROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

CPU ROM CoreSight component identity register 3 (M7_CPUROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

52.10.3 Cortex-M7 CPU ROM table register map and reset values

Table 629. Cortex-M7 CPU ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	M7_CPUROM_MEMTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
0xFD0	M7_CPUROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	M7_CPUROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	0	0	0	1	0	1
0xFE4	M7_CPUROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	0
0xFE8	M7_CPUROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	0	1	0
0xFEC	M7_CPUROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0xFF0	M7_CPUROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	1	0	1
0xFF4	M7_CPUROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	1	0	0	0	0
0xFF8	M7_CPUROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	1
0xFFC	M7_CPUROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	1

52.10.4 Cortex-M7 PPB ROM registers

PPB ROM memory type register (M7_PPBROM_MEMTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory presence
1: System memory is present on this bus

PPB ROM CoreSight peripheral identity register 4 (M7_PPBROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC continuation code

PPB ROM CoreSight peripheral identity register 0 (M7_PPBROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00C7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0xC7: Cortex-M7 PPB ROM table

PPB ROM CoreSight peripheral identity register 1 (M7_PPBROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x4: Cortex-M7 PPB ROM table

PPB ROM CoreSight peripheral identity register 2 (M7_PPBROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

PPB ROM CoreSight peripheral identity register 3 (M7_PPBROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

PPB ROM CoreSight component identity register 0 (M7_PPBROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

PPB ROM CoreSight component identity register 1 (M7_PPBROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

PPB ROM CoreSight component identity register 2 (M7_PPBROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

PPB ROM CoreSight component identity register 3 (M7_PPBROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

52.10.5 Cortex-M7 PPB ROM table register map and reset values

Table 630. Cortex-M7 PPB ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	M7_PPBROM_MEMTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
0xFD0	M7_PPBROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	M7_PPBROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	0	0	0	1	1	1
0xFE4	M7_PPBROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	1	0	0
0xFE8	M7_PPBROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	0	1	1
0xFEC	M7_PPBROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0xFF0	M7_PPBROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	1	0	1
0xFF4	M7_PPBROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	1	0	0	0	0
0xFF8	M7_PPBROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	1
0xFFC	M7_PPBROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	1

52.10.6 Cortex-M7 data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as:

- Watchpoint
- ETM trigger
- PC sampling trigger
- Data address sampling trigger
- Data comparator (comparator 1 only)
- Clock cycle counter comparator (comparator 0 only)

It also contains counters for:

- Clock cycles
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- Number of cycles per instruction
- Interrupt overhead

A DWT comparator compares one of the following with the value held in its DWT_COMP register:

- A data address
- An instruction address
- A data value
- The cycle count value, for comparator 0 only.

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- One or more DWT data trace packets, containing one or more of:
  - The address of the instruction that caused a data access
  - An address offset, bits[15:0] of the data access address
  - The matched data value
- A watchpoint debug event, on either the PC value or the accessed data address
- A CMPMATCH[N] event that signals the match outside the DWT unit

A watchpoint debug event either generates a DebugMonitor exception, or causes the processor to halt execution and enter Debug state.

For more details on how to use the DWT, refer to the Arm®v7-M Architecture Reference Manual *[5]*.

52.10.7 Cortex-M7 DWT registers

DWT control register (M7_DWT_CTRL)

Address offset: 0x000

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	Res.	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA
r	r	r	r	r	r	r	r		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSTINIT[3:0]				POSTRESET[3:0]				CYCCNTENA
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **NUMCOMP[3:0]**: Number of implemented comparators (read-only)
0x4: Four comparators

Bit 27 **NOTRCPKT**: Trace sampling and exception tracing support (read-only)
0: Supported

Bit 26 **NOEXTTRIG**: External match signal, CMPMATCH support (read-only)
0: Supported

Bit 25 **NOCYCCNT**: Cycle counter support (read-only)
0: Supported

Bit 24 **NOPRFCNT**: Profiling counter support (read-only)
0: Supported

Bit 23 Reserved, must be kept at reset value.

Bit 22 **CYCEVTENA**: Enable POSTCNT underflow event counter packet generation
0: Disabled
1: Enabled

Bit 21 **FOLDEVTENA**: Enable folded instruction counter overflow event generation
0: Disabled
1: Enabled

Bit 20 **LSUEVTENA**: Enable LSU counter overflow event generation
0: Disabled
1: Enabled

Bit 19 **SLEEPEVTENA**: Enable sleep counter overflow event generation
0: Disabled
1: Enabled

Bit 18 **EXCEVTENA**: Enable exception overhead counter overflow event generation
0: Disabled
1: Enabled

Bit 17 **CPIEVTENA**: Enable CPI counter overflow event generation
0: Disabled
1: Enabled

Bit 16 **EXCTRCENA**: Enable exception trace generation
0: Disabled
1: Enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PCSAMPLENA**: POSTCNT counter use enable
Enables the use of POSTCNT counter as a timer for Periodic PC sample packet generation.

0: Disabled
1: Enabled

Bits 11:10 **SYNCTAP[1:0]**: Position of synchronization packet counter tap on CYCCNT counter
This selection determines the synchronization packet rate.
0x0: Disabled - no synchronization packets
0x1: Tap at CYCCNT[24]
0x2: Tap at CYCCNT[26]
0x3: Tap at CYCCNT[28]

Bit 9 **CYCTAP**: Position of the POSTCNT tap on the CYCCNT counter
0: Tap at CYCCNT[6]
1: Tap at CYCCNT[10]

Bits 8:5 **POSTINIT[3:0]**: Initial value of the POSTCNT counter
Writes to this field are ignored if POSTCNT counter is enabled (that is, CYCEVTENA or PCSAMPLENA must be reset prior to writing POSTINIT).

Bits 4:1 **POSTRESET[3:0]**: Reload value of the POSTCNT counter.

Bit 0 **CYCCNTENA**: CYCCNT counter enable
0: Disabled
1: Enabled

DWT cycle count register (M7_DWT_CYCCNT)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CYCCNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CYCCNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CYCCNT[31:0]**: Processor clock cycle counter

DWT CPI count register (M7_DWT_CPICNT)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CPICNT[7:0]**: CPI counter

Counts additional cycles required to execute multi-cycle instructions, except those recorded by DWT_LSUCNT, and counts any instruction fetch stalls.

DWT exception count register (M7_DWT_EXCCNT)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EXCCNT[7:0]**: Exception overhead cycle counter

Counts the number of cycles spent in exception processing.

DWT sleep count register (M7_DWT_SLPCNT)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SLEEPCNT[7:0]**: Sleep cycle counter
Counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).

DWT LSU count register (M7_DWT_LSUCNT)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LSUCNT[7:0]**: Load store counter
Counts additional cycles required to execute load and store instructions.

DWT fold count register (M7_DWT_FOLDCNT)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **FOLDCNT[7:0]**: Folded instruction counter
Increments on each instruction that takes 0 cycles.

DWT program counter sample register (M7_DWT_PCSR)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EIASAMPLE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIASAMPLE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **EIASAMPLE[31:0]**: Executed instruction address sample value
Samples the current value of the program counter.

DWT comparator register x (M7_DWT_COMPx)

Address offset: 0x020 + x * 0x10 (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **COMP[31:0]**: Reference value for comparison.

DWT mask register x (M7_DWT_MASKx)

Address offset: 0x024 + x * 0x10 (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **MASK[4:0]**: Comparator mask size
Provides the size of the ignore mask applied to the access address for address range matching by comparator n. A debugger can write 0b11111 to this field and then read the register back to determine the maximum mask size supported.

DWT function register x (M7_DWT_FUNCTx)

Address offset: 0x028 + x * 0x10 (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]			
							r					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAVADDR0[3:0]				DATAVSIZE[1:0]		LINK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match (read-only)
Indicates if a comparator match has occurred since the register was last read.
0: No match
1: Match occurred

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **DATAVADDR1[3:0]**: Comparator number of a second comparator
When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a second comparator to use for linked address comparison.

Bits 15:12 **DATAVADDR0[3:0]**: Comparator number of a comparator
When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a comparator to use for linked address comparison.

Bits 11:10 **DATAVSIZE[1:0]**: Size of required data comparison
For data value matching, specifies the size of the required data comparison.
0x0: Byte
0x1: Half word
0x2: Word
0x3: Reserved

Bit 9 **LINK1ENA**: Support of a second linked comparator (read-only)
Indicates whether the use of a second linked comparator is supported (read-only).
1: Supported

Bit 8 **DATAVMATCH**: Cycle comparison enable
0: Perform address comparison
1: Perform data value comparison

Bit 7 **CYCMATCH**: Cycle count comparison enable on comparator 0
This field is reserved for other comparators.
0: No cycle count comparison
1: Compare DWT_COMP0 with the cycle counter, DWT_CYCCNT

Bit 6 Reserved, must be kept at reset value.

Bit 5 **EMITRANGE**: Data trace address offset packet enable
Enables the generation of data trace address offset packets (containing data address bits 0 to 15)
0: Disabled
1: Enabled

Bit 4 Reserved, must be kept at reset value.

Bits 3:0 **FUNCTION[3:0]**: Action on comparator match
The meaning of this bit field depends on the setting of the DATAVMATCH and CYCMATCH fields. See *[5]*.

DWT CoreSight peripheral identity register 4 (M7_DWT_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

DWT CoreSight peripheral identity register 0 (M7_DWT_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x02: DWT part number

DWT CoreSight peripheral identity register 1 (M7_DWT_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x0: DWT part number

DWT CoreSight peripheral identity register 2 (M7_DWT_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

DWT CoreSight peripheral identity register 3 (M7_DWT_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

DWT CoreSight component identity register 0 (M7_DWT_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

DWT CoreSight component identity register 1 (M7_DWT_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

DWT CoreSight component identity register 2 (M7_DWT_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

DWT CoreSight component identity register 3 (M7_DWT_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

52.10.8 Cortex-M7 DWT register map and reset values

The Cortex-M7 DWT registers are located at address range 0xE0001000 to 0xE0001FFC, on the AHBD.

Table 631. Cortex-M7 DWT register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	M7_DWT_CTRL	NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	Res.	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA	Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSINIT[3:0]				POSTPRESET[3:0]				CYCCNTENA
	Reset value	0	1	0	0	0	0	0	0	-	0	0	0	0	0	0	0	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	M7_DWT_CYCCNT	CYCCNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	M7_DWT_CPICNT	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CPICNT[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x00C	M7_DWT_EXCCNT	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	EXCCNT[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x010	M7_DWT_SLPCNT	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SLEEPCNT[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x014	M7_DWT_LSUCNT	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	LSUCNT[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0

Table 631. Cortex-M7 DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	M7_DWT_FOLDCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x01C	M7_DWT_PCSR	EIASAMPLE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	M7_DWT_COMP0	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	M7_DWT_MASK0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0
0x028	M7_DWT_FUNCT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]				DATAVADDR0[3:0]				DATAVSIZE[1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
	Reset value	-	-	-	-	-	-	-	0	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	-	0	0	0	0
0x02C	Reserved																																
0x030	M7_DWT_COMP1	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	M7_DWT_MASK1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0
0x038	M7_DWT_FUNCT1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]				DATAVADDR0[3:0]				DATAVSIZE[1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
	Reset value	-	-	-	-	-	-	-	0	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	-	0	0	0	0
0x03C	Reserved																																
0x040	M7_DWT_COMP2	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	M7_DWT_MASK2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0
0x048	M7_DWT_FUNCT2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]				DATAVADDR0[3:0]				DATAVSIZE[1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
	Reset value	-	-	-	-	-	-	-	0	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	-	0	0	0	0
0x04C	Reserved																																
0x050	M7_DWT_COMP3	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	M7_DWT_MASK3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0

Table 631. Cortex-M7 DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x058	M7_DWT_FUNCT3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]				DATAVADDR0[3:0]				DATAVSIZE[1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
	Reset value	-	-	-	-	-	-	-	0	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	-	0	0	0	0
0x05C to 0xFCC	Reserved																																
0xFD0	M7_DWT_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	0
0xFD4 to 0xFCC	Reserved																																
0xFE0	M7_DWT_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	1	0
0xFE4	M7_DWT_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	1	0	0	1
0xFE8	M7_DWT_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	0	1	1
0xFEC	M7_DWT_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0xFF0	M7_DWT_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	1	0	1
0xFF4	M7_DWT_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	1	1	0	0	0	0
0xFF8	M7_DWT_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	1
0xFFC	M7_DWT_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	1

52.10.9 Cortex-M7 instrumentation trace macrocell (ITM)

The ITM generates trace information as packets. There are three sources that can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The three sources in decreasing order of priority are:

1. Software trace

   Software can write directly to any of 32 x 32-bit ITM stimuli registers to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the identity of the port, the size of the write access, and the data written into a packet that it writes to a FIFO. The ITM outputs

packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.

2. Hardware trace

   The DWT generates trace packets in response to a data trace event, a PC sample or a performance profiling counter wraparound. The ITM outputs these packets on the trace bus.

3. Local timestamping

   The ITM contains a 21-bit counter clocked by the (pre-divided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset to zero every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.

52.10.10 Cortex-M7 ITM registers

ITM stimulus register x (M7_ITM_STIMx)

Address offset: 0x000 + x * 0x4 (x = 0 to 31)

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMULUS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMULUS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **STIMULUS[31:0]**: Software event packet / FIFOREADY

Write data is output on the trace bus as a software event packet. When reading, bit 0 is a FIFOREADY indicator:

0: Stimulus port buffer is full (or port is disabled)

1: Stimulus port can accept new write data

ITM trace enable register (M7_ITM_TER)

Address offset: 0xE00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMENA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMENA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **STIMENA[31:0]**: Stimulus port enable

Each bit n (31:0) enables the stimulus port associated with the M7_ITM_STIMn register.

0: Port disabled

1: Port enabled

ITM trace privilege registers (M7_ITM_TPR)

Address offset: 0xE40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVMASK[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRIVMASK[3:0]**: Enable unprivileged access to ITM stimulus ports
Each bit controls eight stimulus ports:
0bXXX0: Unprivileged access permitted on ports 0 to 7
0bXXX1: Only privileged access permitted on ports 0 to 7
0bXX0X: Unprivileged access permitted on ports 8 to 15
0bXX1X: Only privileged access permitted on ports 8 to 15
0bX0XX: Unprivileged access permitted on ports 16 to 23
0bX1XX: Only privileged access permitted on ports 16 to 23
0b0XXX: Unprivileged access permitted on ports 24 to 31
0b1XXX: Only privileged access permitted on ports 24 to 31

ITM trace control register (M7_ITM_TCR)

Address offset: 0xE80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]						
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSPRESCALE[1:0]		Res.	Res.	Res.	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
						rw	rw				r	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **BUSY**: ITM busy
Indicates whether the ITM is currently processing events (read-only):
0: Not busy
1: Busy

Bits 22:16 **TRACEBUSID[6:0]**: Identifier for multi-source trace stream formatting
If multi-source trace is in use, the debugger must write a non-zero value to this field. Note: different IDs must be used for each trace source in the system.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **TSPRESCALE[1:0]**: Local timestamp prescale
Prescaler used with the trace packet reference clock The possible values are:
0x0: No prescaling
0x1: Divide by 4
0x2: Divide by 16
0x3: Divide by 64

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SWOENA**: Asynchronous clocking enable for the timestamp counter (read-only)
0: Timestamp counter uses processor clock

Bit 3 **TXENA**: Hardware event packet forwarding enable
Enables forwarding of hardware event packets from the DWT unit to the trace port.
0: Disabled
1: Enabled

Bit 2 **SYNCENA**: Synchronization packet transmission enable
If a debugger sets this bit it must also configure the DWT_CTRL register SYNCTAP field in the DWT for the correct synchronization speed.
0: Disabled
1: Enabled

Bit 1 **TSENA**: Local timestamp generation enable
0: Disabled
1: Enabled

Bit 0 **ITMENA**: ITM enable
0: Disabled
1: Enabled

ITM CoreSight peripheral identity register 4 (M7_ITM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

ITM CoreSight peripheral identity register 0 (M7_ITM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x01: ITM part number

ITM CoreSight peripheral identity register 1 (M7_ITM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x1: ITM part number

ITM CoreSight peripheral identity register 2 (M7_ITM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

ITM CoreSight peripheral identity register 3 (M7_ITM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

ITM CoreSight component identity register 0 (M7_ITM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

ITM CoreSight component identity register 1 (M7_ITM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

ITM CoreSight component identity register 2 (M7_ITM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

ITM CoreSight component identity register 3 (M7_ITM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

52.10.11 Cortex-M7 ITM register map and reset values

The ITM registers are located at address range 0xE0000000 to 0xE0000FFC, on the AHBD.

Table 632. Cortex-M7 ITM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 to 0x07C	M7_ITM_STIM0-31	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x080 to 0xDFC	Reserved																																
0xE00	M7_ITM_TER	STIMENA[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xE04 to 0xE3C	Reserved																																
0x0E40	M7_ITM_TPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVMASK[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
0xE44 to 0xE7C	Reserved																																
0xE80	M7_ITM_TCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	TSPRESCALE[1:0]		Res.	Res.	Res.	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
	Reset value	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	-	-	-	-	-	-	0	0	-	-	-	0	0	0	0	0
0xE84 to 0xFCC	Reserved																																
0xFD0	M7_ITM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	M7_ITM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	1
0xFE4	M7_ITM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	0
0xFE8	M7_ITM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	0	1	1

Table 632. Cortex-M7 ITM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFEC	M7_ITM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0xFF0	M7_ITM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	1	0	1
0xFF4	M7_ITM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	0	0	0	0	0
0xFF8	M7_ITM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	1
0xFFC	M7_ITM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	1

52.10.12 Cortex-M7 breakpoint unit (FPB)

The FPB allows hardware breakpoints to be set. It contains eight comparators which monitor the instruction fetch address and return a breakpoint instruction when a match is detected. The Cortex-M7 FPB does not support flash patch functionality.

52.10.13 Cortex-M7 FPB registers

FPB control register (M7_FPB_CTRL)

Address offset: 0x000

Reset value: 0x1000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NUM_CODE[6:4]			NUM_LIT[3:0]				NUM_CODE[3:0]				Res.	Res.	KEY	ENABLE
	r	r	r	r	r	r	r	r	r	r	r			rw	rw

Bits 31:28 **REV[3:0]**: Flash patch breakpoint architecture revision.
0x1: Flash patch breakpoint version 2.

Bits 27:15 Reserved, must be kept at reset value.

Bits 14:12 **NUM_CODE[6:4]**: Instruction address comparator number field, three MSBs
This read-only field holds the three MSBs of the number of instruction address comparators supported.
0x0: the MSBs of the number are all 0

Bits 11:8 **NUM_LIT[3:0]**: Number of literal address comparators supported (read-only).
0x0: No literal comparators supported.

Bits 7:4 **NUM_CODE[3:0]**: Instruction address comparator number field, four LSBs
This read-only field holds the four LSBs of the number of instruction address comparators supported.
0x8: 8 instruction comparators supported

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **KEY**: Write protect key
A write to M7_FPB_CTRL register is ignored if this bit is not set to 1.

Bit 0 **ENABLE**: FPB enable
0: Disable
1: Enable

FPB comparator registers (M7_FPB_COMPx)

Address offset: 0x008 + x * 0x4 (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BPADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BPADDR[15:1]															BE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:1 **BPADDR[31:1]**: Breakpoint address.
This bitfield specifies the breakpoint instruction address.

Bit 0 **BE**: Breakpoint enable
0: Disabled
1: Enabled

FPB CoreSight peripheral identity register 4 (M7_FPB_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

FPB CoreSight peripheral identity register 0 (M7_FPB_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 000E

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x0E: FPB part number

FPB CoreSight peripheral identity register 1 (M7_FPB_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x0: FPB part number

FPB CoreSight peripheral identity register 2 (M7_FPB_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

FPB CoreSight peripheral identity register 3 (M7_FPB_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

FPB CoreSight component identity register 0 (M7_FPB_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

FPB CoreSight component identity register 1 (M7_FPB_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

FPB CoreSight component identity register 2 (M7_FPB_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

FPB CoreSight component identity register 3 (M7_FPB_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

52.10.14 Cortex-M7 FPB register map and reset values

The Cortex-M7 FPB registers are located at address range 0xE0002000 to 0xE0002FFC.

Table 633. Cortex-M7 FPB register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	M7_FPB_CTRL	REV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUM_CODE[6:4]			NUM_LIT[3:0]				NUM_CODE[3:0]				Res.	Res.	KEY	ENABLE
	Reset value	0	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	1	0	0	0	-	-	0	0
0x004	Reserved																																
0x008 to 0x020	M7_FPB_COMP0 to M7_FPB_COMP7	BPADDR[31:1]																															BE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024 to 0xFCC	Reserved																																
0xFD0	M7_FPB_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	M7_FPB_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	1	1	0
0xFE4	M7_FPB_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	0
0xFE8	M7_FPB_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	0	1	1
0xFEC	M7_FPB_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0xFF0	M7_FPB_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	1	0	1
0xFF4	M7_FPB_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	0	0	0	0	0
0xFF8	M7_FPB_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	1
0xFFC	M7_FPB_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	1

52.10.15 Cortex-M7 embedded trace macrocell (ETM)

The Cortex-M7 ETM is a CoreSight™ component closely coupled to the CPU. The ETM generates trace packets that allow the execution of the Cortex-M7 core to be traced. In the STM32H7, the ETM is configured for instruction trace only, so data accesses are not included in the trace information.

The ETM receives information from the CPU over the processor trace interface, including:

- The number of instructions executed in the same cycle
- Changes in program flow
- The current processor instruction state
- The memory address locations accessed by load and store instructions
- The type, direction and size of a transfer
- Condition code information
- Exception information
- Wait for interrupt state information

For more information, refer to the Arm® CoreSight™ ETM™-M7 technical reference manual *[6]*.

52.10.16 Cortex-M7 ETM registers

ETM programming control register (M7_ETM_PRGCTL)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EN**: Trace program enable
0: Trace unit is disabled
1: Trace unit is enabled

ETM processor select control register (M7_ETM_PROCSEL)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROCSEL
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PROCSEL**: Processor select

This field has no effect since only the Cortex-M7 uses this ETM.

ETM status register (M7_ETM_STAT)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMSTABLE	IDLE
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PMSTABLE**: Programmers model stable

Indicates whether the ETM registers are stable and can be read.

0: Registers are not stable

1: Registers are stable

Bit 0 **IDLE**: Trace unit inactive

0: ETM is not idle

1: ETM is idle

ETM trace configuration register (M7_ETM_CONFIG)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DV	DA
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RS	TS	COND[2:0]			Res.	Res.	Res.	CCI	BB	INSTP0[1:0]		Res.
			rw	rw	rw	rw	rw				rw	rw	r	r	

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DV**: Data value tracing (read-only)
0: Disabled

Bit 16 **DA**: Data address tracing (read-only)
0: Disabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **RS**: Return stack enable
0: Disabled
1: Enabled

Bit 11 **TS**: Global timestamp tracing
Global timestamp is not supported on this device.
0: Disabled
1: Reserved

Bits 10:8 **COND[2:0]**: Conditional instruction tracing
0x0: Conditional instruction tracing disabled
0x1: Conditional load instructions are traced
0x2: Conditional store instructions are traced
0x3: Conditional load and store instructions are traced
0x7: All conditional instructions are traced
Other: Reserved

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CCI**: Cycle counting in instruction trace
0: Disabled
1: Enabled

Bit 3 **BB**: Branch broadcast mode
0: Disabled
1: Enabled

Bits 2:1 **INSTP0[1:0]**: Determines which instructions are P0 instructions (read-only)
0x0: Only branches are P0 instructions

Bit 0 Reserved, must be kept at reset value.

ETM event control 0 register (M7_ETM_EVENTCTL0)

Only accepts writes when trace unit is disabled

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE1	Res.	Res.	Res.	SEL1[3:0]				TYPE0	Res.	Res.	Res.	SEL0[3:0]			
rw				rw	rw	rw	rw	rw				rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TYPE1**: Resource type for event 1
0: Single selected resource
1: Boolean combined resource pair

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:8 **SEL1[3:0]**: Resource / Boolean combined resource pair, for event 1
When TYPE1 is 0, selects a single selected resource from 0-15 defined by bits[3:0]
When TYPE1 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0]

Bit 7 **TYPE0**: Resource type for event 0
0: Single selected resource
1: Boolean combined resource pair

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL0[3:0]**: Resource / Boolean combined resource pair for event 0
When TYPE0 is 0, selects a single selected resource from 0-15 defined by bits[3:0]
When TYPE0 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0]

ETM event control 1 register (M7_ETM_EVENTCTL1)

Only accepts writes when trace unit is disabled

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[3:0]			
			rw	rw								rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LPOVERRIDE**: Low power state behavior override
0: Low power state normal behavior
1: Entry to low power state does not affect resources and event trace generation

Bit 11 **ATB**: ATB trigger enable
0: Disabled
1: Enabled

Bits 10:4 Reserved, must be kept at reset value.

Bits 3:0 **INSTEN[3:0]**: Instruction trace event element enable
Each bit corresponds to an event:

0bXXX0: Event 0 does not cause an event element
0bXXX1: Event 0 causes an event element
0bXX0X: Event 1 does not cause an event element
0bXX1X: Event 1 causes an event element
0bX0XX: Event 2 does not cause an event element
0bX1XX: Event 2 causes an event element
0b0XXX: Event 3 does not cause an event element
0b1XXX: Event 3 causes an event element

ETM stall control register (M7_ETM_STALLCTL)

Only accepts writes when trace unit is disabled

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	DSTALL	ISTALL	Res.	Res.	Res.	Res.	LEVEL[1:0]		Res.	Res.
						rw	rw					rw	rw		

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **DSTALL**: Stall processor based on data trace buffer space
0: Do not stall processor
1: Stall processor

Bit 8 **ISTALL**: Stall processor based on instruction trace buffer space
0: Do not stall processor
1: Stall processor

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 **LEVEL[1:0]**: Stalling threshold level
A low level minimizes the amount of processor stalling, with a higher risk of FIFO overflow. A high level minimizes the risk of FIFO overflow but increases the amount of processor stalling.

Bits 1:0 Reserved, must be kept at reset value.

ETM global timestamp control register (M7_ETM_TSCTL)

Only accepts writes when trace unit is disabled

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPE	Res.	Res.	Res.	SEL[3:0]			
								rw				rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **TYPE**: Resource type for time stamp insertion
0: Single selected resource
1: Boolean combined resource pair

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL[3:0]**: Resource / Boolean combined resource pair
When TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0]
When TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0]

ETM synchronization period register (M7_ETM_SYNCP)

Address offset: 0x034

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIOD[4:0]				
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PERIOD[4:0]**: Trace bytes between synchronization requests
Defines the number of bytes of trace information between trace synchronization requests.
0xA: 1024 bytes

ETM cycle count control register (M7_ETM_CCCTL)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	THRESHOLD[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **THRESHOLD[11:0]**: Threshold value for instruction trace cycle counting
The threshold represents the minimum interval between cycle count trace packets.
0x0: Reserved
Other: Threshold

ETM trace ID register (M7_ETM_TRACEID)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **TRACEID[6:0]**: Trace ID
0x00: Reserved
0x01 to 0x6F: Valid ID
0x70 to 0x7F: Reserved

ETM ViewInst main control register (M7_ETM_VICTL)

Address offset: 0x080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXLEVEL_S3	Res.	Res.	EXLEVEL_S0
												rw			rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRCERR	TRCRESET	SSSTATUS	Res.	TYPE	Res.	Res.	Res.	SEL[3:0]			
				rw	rw	rw		rw				rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **EXLEVEL_S3**: Trace disable, exception level 3
Disables tracing in the specified exception level in Secure state for exception level 3.
0: Enable ViewInst in this exception level
1: Disable ViewInst in this exception level

Bits 18:17 Reserved, must be kept at reset value.

Bit 16 **EXLEVEL_S0**: Trace disable, exception level 0
Disables tracing in the specified exception level in Secure state for exception level 0.
0: Enable ViewInst in this exception level
1: Disable ViewInst in this exception level

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **TRCERR**: Tracing of system error exception
Selects whether a system error exception must always be traced.

0: System error exception is traced only if the instruction or exception immediately before the system error exception is traced
1: System error exception is always traced regardless of the value of ViewInst

Bit 10 **TRCRESET**: Tracing of reset exception
Selects whether a reset exception must always be traced.

0: Reset exception is traced only if the instruction or exception immediatelybefore the reset exception is traced
1: Reset exception is always traced regardless of the value of ViewInst

Bit 9 **SSSTATUS**: Current status of the start/stop logic
0: Stop state
1: Started state

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TYPE**: Resource type
0: Single selected resource
1: Boolean combined resource pair

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL[3:0]**: Resource / Boolean combined resource pair
When TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0]
When TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0]

ETM ViewInst start/stop control register (M7_ETM_VISSCTL)

Address offset: 0x088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	START[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **STOP[7:0]**: Single address comparator selector to stop trace

Defines the single address comparators to stop trace with the ViewInst Start/Stop control. One bit is provided for each implemented single address comparator.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **START[7:0]**: Single address comparator selector to start trace

Defines the single address comparators to start trace with the ViewInst Start/Stop control. One bit is provided for each implemented single address comparator.

ETM ViewInst start/stop processor comparator control register (M7_ETM_VIPCSSCTL)

Address offset: 0x08C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	START[3:0]			
												rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **STOP[3:0]**: Processor comparator input selector to stop trace

Selects which processor comparator inputs are in use with ViewInst start-stop control, for the purpose of stopping trace. One bit is provided for each processor comparator input.

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **START[3:0]**: Processor comparator input selector to start trace

Selects which processor comparator inputs are in use with ViewInst start-stop control, for the purpose of starting trace. One bit is provided for each processor comparator input.

ETM counter reload value register (M7_ETM_CNTRLDV)

Address offset: 0x140

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VALUE[15:0]**: Counter reload value

This value is loaded into the counter each time the reload event occurs.

ETM ID register 8 (M7_ETM_IDR8)

Address offset: 0x180

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MAXSPEC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAXSPEC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **MAXSPEC[31:0]**: Maximum speculation depth

Indicates the maximum speculation depth of the instruction trace stream. This is the maximum number of P0 elements that have not been committed in the trace stream at any one time.

0x2: Maximum trace speculation depth is 2

ETM ID register 9 (M7_ETM_IDR9)

Address offset: 0x184

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP0KEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP0KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP0KEY[31:0]**: Number of P0 right-hand keys used

0x0: No P0 keys used in instruction trace only configuration

ETM ID register 10 (M7_ETM_IDR10)

Address offset: 0x188

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1KEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP1KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1KEY[31:0]**: Total number of P1 right-hand keys

Indicates the total number of P1 right-hand keys, including normal and special keys.

0x0: No P1 keys used in instruction trace only configuration

ETM ID register 11 (M7_ETM_IDR11)

Address offset: 0x18C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1SPC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP1SPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1SPC[31:0]**: Total number of special P1 right-hand keys used
0x0: No special P1 keys used

ETM ID register 12 (M7_ETM_IDR12)

Address offset: 0x190

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCONDKEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMCONDKEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMCONDKEY[31:0]**:
Indicates the total number of conditional instruction right-hand keys, including normal and special keys.
0x1: One conditional instruction right hand-key implemented

ETM ID register 13 (M7_ETM_IDR13)

Address offset: 0x194

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCONDSPC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMCONDSPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMCONDSPC[31:0]**: Number of special conditional instruction right-hand keys
0x0: No special conditional instruction right hand-keys implemented

ETM implementation specific register 0 (M7_ETM_IMSPEC0)

Address offset: 0x1C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUPPORT[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **SUPPORT[3:0]**: Support for implementation specific extensions
0x0: No implementation specific extensions are supported

ETM ID register 0 (M7_ETM_IDR0)

Address offset: 0x1E0

Reset value: 0x0C00 1EE1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	COMMOPT	TSSIZE[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	QSUPP[1]
		r	r	r	r	r	r								r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QSUPP[0]	Res.	CONDTYPE[1:0]		NUMEVENT[1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	Res.	Res.	Res.	Res.	Res.
r		r	r	r	r	r		r	r	r					

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **COMMOPT**: Meaning of the commit field in some packets
0: Commit mode 0

Bits 28:24 **TSSIZE[4:0]**: Global timestamp size
0x08: Maximum of 64-bit global timestamp implemented

Bits 23:17 Reserved, must be kept at reset value.

Bits 16:15 **QSUPP[1:0]**: Q element support
0x0: Q elements not supported

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **CONDTYPE[1:0]**: Conditional result tracing type
0x1: APSR condition flag values traced

Bits 11:10 **NUMEVENT[1:0]**: Number of events supported in the trace
0x1: Two events supported for instruction only configuration

Bit 9 **RETSTACK**: Return stack support
1: Two entry return stack supported

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TRCCCI**: Support for cycle counting in the instruction trace
1: Cycle counting in the instruction trace is implemented

Bit 6 **TRCCOND**: Support for conditional instruction tracing
1: Conditional instruction trace is implemented

Bit 5 **TRCBB**: Support for branch broadcast tracing
1: Branch broadcast trace is implemented

Bits 4:0 Reserved, must be kept at reset value.

ETM ID register 1 (M7_ETM_IDR1)

Address offset: 0x1E4

Reset value: 0x4100 F401

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRCARCHMAJ[3:0]				TRCARCHMIN[3:0]				REVISION[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DESIGNER[7:0]**: Trace unit designer entity
0x41: Arm®

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:8 **TRCARCHMAJ[3:0]**: Major trace unit architecture version number
0x4: ETM v4

Bits 7:4 **TRCARCHMIN[3:0]**: Minor trace unit architecture version number
0x0: Minor version 0

Bits 3:0 **REVISION[3:0]**: Implementation revision number
0x1: Rev 1

ETM ID register 2 (M7_ETM_IDR2)

Address offset: 0x1E8

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:1]			
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DASIZE[0]	VMIDSIZE[4:0]					CIDSIZE[4:0]					IASIZE[4:0]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **CCSIZE[3:0]**: Cycle counter size
Indicates the size of the cycle counter in bits minus 12.
0x0: Cycle counter is 12 bits

Bits 24:20 **DVSIZE[4:0]**: Data value size in bytes
0x0: Data value size is not supported in instruction only configuration

Bits 19:15 **DASIZE[4:0]**: Data address size in bytes
0x0: Data address size is not supported in instruction only configuration

Bits 14:10 **VMIDSIZE[4:0]**: Virtual machine ID size
0x0: Virtual machine ID tracing is not implemented

Bits 9:5 **CIDSIZE[4:0]**: Context ID size
0x0: Context ID tracing is not implemented

Bits 4:0 **IASIZE[4:0]**: Instruction address size
0x4: 32-bit maximum address size

ETM ID register 3 (M7_ETM_IDR3)

Address offset: 0x1EC

Reset value: 0x0509 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NOOVERFLOW	NUMPROC[2:0]			SYSSTALL	STALLCTL	SYNCPR	TRCERR	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CCITMIN[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **NOOVERFLOW**: Support of NOOVERFLOW
Indicates whether the NOOVERFLOW of trace stall control is implemented.
0: Not implemented

Bits 30:28 **NUMPROC[2:0]**: Number of processors available for tracing
0x0: Only one processor can be traced

Bit 27 **SYSSTALL**: System support for stall control of the processor
0: Not supported

Bit 26 **STALLCTL**: Stall control support
1: Trace stall control (TRCSTALLCTLR) is implemented

Bit 25 **SYNCPR**: Trace synchronization period support
0: TRCSYNCPR is read-only for instruction trace only configuration; the trace synchronization period is fixed

Bit 24 **TRCERR**: Support of TRCVICTLR.TRCERR
Indicates whether TRCVICTLR.TRCERR is implemented.
0x4: 32-bit maximum address size

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **EXLEVEL_S[3:0]**: Support of privilege levels
Privilege levels are implemented; one bit for each level.
0x9: Privilege levels Thread and Handler are implemented

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **CCITMIN[11:0]**: Instruction trace cycle counting minimum threshold
0x4: Minimum threshold is 4 instruction trace cycle

ETM ID register 4 (M7_ETM_IDR4)

Address offset: 0x1F0

Reset value: 0x0001 4000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMVMIDC[3:0]				NUMCIDC[3:0]				NUMSSCC[3:0]				NUMRSPAIR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMPC[3:0]				Res.	Res.	Res.	SUPPADC	NUMDVC[3:0]				NUMACPAIRS[3:0]			
r	r	r	r				r	r	r	r	r	r	r	r	r

Bits 31:28 **NUMVMIDC[3:0]**: Number of Virtual Machine ID comparators implemented
0x0: None

Bits 27:24 **NUMCIDC[3:0]**: Number of Context ID comparators implemented
0x0: None

Bits 23:20 **NUMSSCC[3:0]**: Number of single-shot comparator controls implemented
0x0: None

Bits 19:16 **NUMRSPAIR[3:0]**: Number of resource selection pairs implemented
0x1: None

Bits 15:12 **NUMPC[3:0]**: Number of processor comparator inputs implemented
0x4: Four

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SUPPADC**: Support of data address comparisons
0: Not implemented

Bits 7:4 **NUMDVC[3:0]**: Number of data value comparators implemented
0x0: None

Bits 3:0 **NUMACPAIRS[3:0]**: Number of address comparator pairs implemented.
0x0: None

ETM ID register 5 (M7_ETM_IDR5)

Address offset: 0x1F4

Reset value: 0x90C7 0402

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE[2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE[5:0]					
r	r	r	r	r	r	r		r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NUMEXTINSEL[2:0]			NUMEXTIN[8:0]								
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **REDFUNCNTR**: Support of reduced function counter
1: Implemented

Bits 30:28 **NUMCNTR[2:0]**: Number of counters implemented
0x1: One counter implemented

Bits 27:25 **NUMSEQSTATE[2:0]**: Number of sequencer states implemented
0x0: None

Bit 24 Reserved, must be kept at reset value.

Bit 23 **LPOVERRIDE**: Support of low-power state override
1: Implemented

Bit 22 **ATBTRIG**: Support of ATB trigger
1: Implemented

Bits 21:16 **TRACEIDSIZE[5:0]**: Number of trace ID bits
0x07: Seven-bit trace ID implemented.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:9 **NUMEXTINSEL[2:0]**: Number of implemented external input selectors
0x2: Two external input selectors implemented

Bits 8:0 **NUMEXTIN[8:0]**: Number of implemented external inputs
0x2: Two external inputs implemented

ETM resource selection register 2 (M7_ETM_RSCTL2)

Address offset: 0x208

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAIRINV	INV	Res.	GROUP[2:0]		
										rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **PAIRINV**: Inversion of result of a combined pair of resources
0: Not inverted
1: Inverted

Bit 20 **INV**: Inversion of the selected resources
0: Not inverted
1: Inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: Selects a group of resources

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SELECT[7:0]**: Resource selector from desired group
Selects one or more resources from the desired group. One bit is provided per resource from the group.

ETM resource selection register 3 (M7_ETM_RSCTL3)

Address offset: 0x20C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INV	Res.	GROUP[2:0]		
											rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **INV**: Inversion of the selected resources
0: Not inverted
1: Inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: Selects a group of resources

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SELECT[7:0]**: Resource selector from desired group
Selects one or more resources from the desired group. One bit is provided per resource from the group.

ETM single-shot comparator control register 0 (M7_ETM_SSCC0)

Address offset: 0x280

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RST**: Single-shot comparator resource reset enable

Enables the single-shot comparator resource to be reset when it occurs, which enables another comparator match to be detected.

0: Disabled

1: Reset enabled; multiple matches can occur

Bits 23:0 Reserved, must be kept at reset value.

ETM single-shot comparator status register 0 (M7_ETM_SSCS0)

Address offset: 0x2A0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DV	DA	INST
													r	r	r

Bit 31 **STATUS**: Single-shot status

This indicates whether any of the selected comparators have matched. If SSCC0.RST is set to 0, the STATUS bit must be written with 0 in order to enable single-shot comparator control.

0: No match occurred

1: Match has occurred at least once.

Bits 30:3 Reserved, must be kept at reset value.

Bit 2 **DV**: Data value comparator support

0: Single-shot data value comparisons not supported

Bit 1 **DA**: Data address comparator support

0: Single-shot data address comparisons not supported

Bit 0 **INST**: Instruction address comparator support

1: Single-shot instruction address comparisons supported

ETM single-shot processor comparator input control register (M7_ETM_SSPCIC0)

Address offset: 0x2C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PC[7:0]**: Comparator input selector for single-shot control

Selects one or more processor comparator inputs for single-shot control. One bit is provided for each processor comparator input.

ETM power-down control register (M7_ETM_PDC)

Address offset: 0x310

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU	Res.	Res.	Res.
												r			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **PU**: Power up request

Request to maintain power to the ETM and access to the trace registers.

0: Power not requested

1: Power requested

Bits 2:0 Reserved, must be kept at reset value.

ETM power-down status register (M7_ETM_PDS)

Address offset: 0x314

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STICKYPD	POWER
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **STICKYPD**: Sticky power-down state

This bit is set to 1 when power to the ETM registers is removed, to indicate that programming state has been lost. It is cleared after a read of the TRCPDSR.

0: Trace register power uninterrupted since the last read of PDS register

1: Trace register power interrupted since the last read of PDS register

Bit 0 **POWER**: ETM powered up

1: ETM is powered up; all registers are accessible

ETM claim tag set register (M7_ETM_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:

0000: No effect

XXX1: Set bit 0

XX1X: Set bit 1

X1XX: Set bit 2

1XXX: Set bit 3

Read:

0xF: Indicates there are four bits in claim tag

ETM claim tag clear register (M7_ETM_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits

Write:
0000: No effect
XXX1: Clear bit 0
XX1X: Clear bit 1
X1XX: Clear bit 2
1XXX: Clear bit 3
Read: Returns current value of claim tag

ETM lock access register (M7_ETM_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: ETM register write access

Enables write access to some ETM registers by the processor (debuggers do not need to unlock the component)
0xC5ACCE55: Enable write access
Other values: Disable write access

ETM lock status register (M7_ETM_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the M7_ETM_LAR register
0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock
This bit always returns zero when read by an external debugger.
0: Write access is permitted
1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.
0: No lock control mechanism exists
1: Lock control mechanism is implemented

ETM authentication status register (M7_ETM_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x2: Disabled
0x3: Enabled

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x2: Disabled
0x3: Enabled

ETM CoreSight device architecture register (M7_ETM_DEVARCH)

Address offset: 0xFBC

Reset value: 0x4770 4A13

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: Component architect
0x23B: Arm®

Bit 20 **PRESENT**: Indicates the presence of this register
1: Present

Bits 19:16 **REVISION[3:0]**: Architecture revision
0x0: Rev 0

Bits 15:0 **ARCHID[15:0]**: Architecture ID
0x4A13: ETMv4 component

ETM CoreSight device type identity register (M7_ETM_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0013

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Device sub-type identifier
0x1: Processor trace

Bits 3:0 **MAJORTYPE[3:0]**: Device main type identifier
0x3: Trace source

ETM CoreSight peripheral identity register 4 (M7_ETM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

ETM CoreSight peripheral identity register 0 (M7_ETM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0075

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x75: ETM part number

ETM CoreSight peripheral identity register 1 (M7_ETM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: ETM part number

ETM CoreSight peripheral identity register 2 (M7_ETM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x1: rev 1

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

ETM CoreSight peripheral identity register 3 (M7_ETM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

ETM CoreSight component identity register 0 (M7_ETM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

ETM CoreSight component identity register 1 (M7_ETM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: Debug component with CoreSight-compatible registers

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

ETM CoreSight component identity register 2 (M7_ETM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

ETM CoreSight component identity register 3 (M7_ETM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

52.10.17 Cortex-M7 ETM register map and reset values

The ETM registers are accessed by the debugger via the Cortex-M7 PPB, at address range 0xE0041000 to 0xE0041FFC.

Table 634. Cortex-M7 ETM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	M7_ETM_PRGCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
0x008	M7_ETM _PROCSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROCSEL[2:0]		
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0
0x00C	M7_ETM_STAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMSTABLE	IDLE
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0
0x010	M7_ETM_CONFIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DV	DA	Res.	Res.	Res.	RS	TS	COND[2:0]			VMID	CID	Res.	CCI	BB	INSTP0[1:0]		Res.
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-	0	0	0	0	0	0	0	-	0	0	0	0	1
0x014 to 0x01C	Reserved																																
0x020	M7_ETM _EVENTCTL0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPE1	Res.	Res.	Res.	SEL1[3:0]				TYPE0	Res.	Res.	Res.	SEL0[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	0	0	0	0	0	-	-	-	0	0	0	0
0x024	M7_ETM _EVENTCTL1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-	-	-	-	-	0	0	0	0
0x02C	M7_ETM _STALLCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSTALL	ISATLL	Res.	Res.	Res.	Res.	LEVEL[1:0]		Res.	Res.
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-	-	0	0	-	-
0x030	M7_ETM_TSCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPE	Res.	Res.	Res.	SEL[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	0	0	0	0
0x034	M7_ETM_SYNCP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIOD[4:0]				
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	0	1	0
0x038	M7_ETM_CCCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THRESHOLD[11:0]											
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	Reserved																																

Table 634. Cortex-M7 ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	M7_ETM_TRACEID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0
0x044 to 0x07C	Reserved																																
0x080	M7_ETM_VICTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXLEVEL_S3	Res.	Res.	EXLEVEL_S0	Res.	Res.	Res.	Res.	TRCERR	TRCRESET	SSSTATUS	Res.	TYPE	Res.	Res.	Res.	SEL[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	0	-	-	-	-	0	0	0	-	0	-	-	-	0	0	0	0
0x084	Reserved	Reserved																															
0x088	M7_ETM_VISSCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	START[7:0]							
	Reset value	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x08C	M7_ETM _VIPCSSCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	START[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
0x090 to 0x13C	Reserved																																
0x140	M7_ETM _CNTRLDV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VALUE[15:0]															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x144 to 0x17C	Reserved																																
0x180	M7_ETM_IDR8	MAXSPEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x184	M7_ETM_IDR9	NUMP0KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x188	M7_ETM_IDR10	NUMP1KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18C	M7_ETM_IDR11	NUMP1SPC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x190	M7_ETM_IDR12	NUMCONDKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x194	M7_ETM_IDR13	NUMCONDSPC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x198 to 0x1BC	Reserved																																
0x1C0	M7_ETM_IMSPEC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUPPORT[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
0x1E0	M7_ETM_IDR0	Res.	Res.	COMMOPT	TSSIZE[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	QSUPP[1:0]		CONDTYPE[1:0]		NUMEVENT[1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	Res.	Res.	Res.	Res.	Res.		
	Reset value	-	-	0	0	1	1	0	0	-	-	-	-	-	-	-	0	0	-	0	1	1	1	1	-	1	1	1	-	-	-	-	1

Table 634. Cortex-M7 ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1E4	M7_ETM_IDR1	DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCARCHMAJ[3:0]				TRCARCHMIN[3:0]				REVISION[3:0]			
	Reset value	0	1	0	0	0	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	0	1	0	0	0	0	0	0	0	0	0	1
0x1E8	M7_ETM_IDR2	Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:0]					VMIDSIZE[4:0]					CIDSIZE[4:0]					IASIZE[4:0]			
	Reset value	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0x1EC	M7_ETM_IDR3	NOOVERFLOW	NUMPROC[2:0]		SYSSTALL	STALLCTL	SYNCPR	TRCERR		Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]				Res.	Res.	Res.	Res.	CCITMIN[11:0]											
	Reset value	0	0	0	0	0	1	0	1	-	-	-	-	1	0	0	1	-	-	-	-	0	0	0	0	0	0	0	0	0	1	0	0
0x1F0	M7_ETM_IDR4	NUMVMIDC[3:0]				NUMCIDC[3:0]				NUMSSCC[3:0]				NUMRSPAIR[3:0]				NUMPC[3:0]				Res.	Res.	Res.	SUPPDAC	NUMDVC[3:0]				NUMACPAIRS[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	-	-	-	0	0	0	0	0	0	0	0	0
0x1F4	M7_ETM_IDR5	REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE[2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE[5:0]						Res.	Res.	Res.	Res.	NUMEXTINSEL[2:0]		NUMEXTIN[8:0]									
	Reset value	1	0	0	1	0	0	0	-	1	1	0	0	0	1	1	1	-	-	-	-	0	1	0	0	0	0	0	0	0	1	0	
0x1F8 to 0x204	Reserved																																
0x208	M7_ETM_RSCTL2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAIRINV	INV	Res.	GROUP[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	0	0	-	0	0	0	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x20C	M7_ETM_RSCTL3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INV	Res.	GROUP[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	0	-	0	0	0	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x210 to 0x27C	Reserved																																
0x280	M7_ETM_SSCC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x284 to 0x29C	Reserved																																

Table 634. Cortex-M7 ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2A0	M7_ETM_SSCS0	STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DV	DA	INST
	Reset value	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	1
0x2A4 to 0x2BC	Reserved																																
0x2C0	M7_ETM_SSPCIC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0x2C4 to 0x30C	Reserved																																
0x310	M7_ETM_PDC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU	Res.	Res.	Res.
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-
0x314	M7_ETM_PDS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STICKYPD	POWER
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1
0x318 to 0xF9C	Reserved																																
0xFA0	M7_ETM _CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1
0xFA4	M7_ETM _CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
0xFA8to 0xFAC	Reserved																																
0xFB0	M7_ETM_LAR	ACCESS_W[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xFB4	M7_ETM_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	1
0xFB8	M7_ETM _AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	0	1	0
0xFBC	M7_ETM _DEVARCH	ARCHITECT[10:0]											PRESENT	REVISION[3:0]				ARCHID[15:0]															
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	1	0	0	1	1
0xFC0to 0xFC8	Reserved																																

Table 634. Cortex-M7 ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	M7_ETM _DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1
0xFD0	M7_ETM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	M7_ETM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	1	1	0	1	0	1
0xFE4	M7_ETM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	1	0	0	1
0xFE8	M7_ETM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	1	1	0	1	1
0xFEC	M7_ETM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
0xFF0	M7_ETM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	1	1	0	1
0xFF4	M7_ETM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	0	1	0	0	0	0
0xFF8	M7_ETM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	1	0	1
0xFFC	M7_ETM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	1

52.10.18 Cortex-M7 cross trigger interface (CTI)

See *Section 52.6.4*.

52.11 References for debug infrastructure

1. IHI 0031C (ID080813) - Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, Issue C
2. DDI 0480F (ID100313) - Arm® CoreSight™ SoC-400 r3p2 Technical Reference Manual, Issue G
3. DDI 0461B (ID010111) - Arm® CoreSight™ Trace Memory Controller r0p1 Technical Reference Manual, Issue B
4. DDI 0314H - Arm® CoreSight™ Components Technical Reference Manual, Issue H
5. DDI 0403D (ID100710) - Arm®v7-M Architecture Reference Manual, Issue E.b
6. DDI 0494-2a (ID062813) - Arm® CoreSight™ ETM™-M7 r0p1 Technical Reference Manual, Issue D

53 Device electronic signature

The electronic signature is stored in the flash memory area. It can be read using the JTAG/SWD or the CPU. It contains factory-programmed identification data that allow the user firmware or other external devices to automatically match its interface to the characteristics of the STM32H7Pxxx microcontrollers.

53.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

- For use as serial numbers (for example USB string serial numbers or other end applications)
- For use as security keys in order to increase the security of code in flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal flash memory
- To activate secure boot processes, etc.

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits can never be altered by the user.

The 96-bit unique device identifier can also be read in single bytes/half-words/words in different ways and then be concatenated using a custom algorithm.

Base address: 0x08FF F800

Address offset: 0x00

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **U_ID[31:0]:** X and Y coordinates on the wafer

Address offset: 0x04

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **U_ID[63:40]:** LOT_NUM[23:0]
Lot number (ASCII encoded)

Bits 7:0 **U_ID[39:32]:** Wafer number (8-bit unsigned number)

Address offset: 0x08

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **U_ID[95:64]:** LOT_NUM[55:24]
Lot number (ASCII encoded)

53.2 Package data register

Base address: 0x08FF F80C

Address offset: 0x00

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKG_CODE[4:0]				
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	FLASH_SIZE[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **PKG_CODE[4:0]**: Package information code

This bit-field indicates the device package type.

0x00: Reserved
0x01: Reserved
0x02: LQFP100
0x03: TFBGA100
0x04: LQFP128
0x05: LQFP144
0x06: LQFP176
0x07: UFBGA176+25
0x08: Reserved
0x09: Reserved
0x0A: LQFP100 SMPS
0x0B: TFBGA100 SMPS
0x0C: LQFP128 SMPS
0x0D: LQFP144 SMPS
0x0E: LQFP SMPS
0x0F: UFBGA176+25 SMPS
0x1A: LQFP100 SMPS EP
0x1D: LQFP144 SMPS EP
Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **FLASH_SIZE[11:0]**: Flash memory size

0x400: 1 Mbyte
0x800: 2 Mbyte
Others: Reserved

54 Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on *www.st.com* for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers must re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers must not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

55 Revision history

Table 635. Document revision history

Date	Revision	Changes
07-Oct-2026	1	Initial release.

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers’ market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to *www.st.com/trademarks*. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2026 STMicroelectronics – All rights reserved